ESEIAAT

Final Bachelor Thesis

Universitat Politècnica de Catalunya

Aerospace technology bachelor degree

Study on the determination of the field of view of the ASIM instruments aboard of the ISS

Author:
Paulino Gil Mora

Director:
Joan Montanya Puig

Delivery date: 10/06/2019

Acknowledgments

Once I have finished writing this TFG, I want to express my gratitude to all the lightning research group (LRG) team for their patience and help. Moreover, to Joan Montanya, my TFG coordinator. I wish that in the future, the work developed during this 4 months of TFG and almost in the previously year during my internship, could be very useful for the researchers.

After that, I will to thank my father and mother for their patience. Since I was a kid, we got an objective. Finally, it is done.

Contents

1 List of figures **iv**

2 List of tables **vi**

3 Acronyms **vii**

4 Abstract **1**

5 Statement of purpose authority and responsibility **2**

6 Aim **3**

7 Scope **4**

8 Requirements **5**

9 Justification **6**

10 State of the art: ASIM and ISS-LIS **7**
10.1 ASIM introduction 7
10.2 LIS 8
10.3 MMIA 11

11 ISS characteristics **12**
11.1 ISS orbit information 12
11.2 Orbit monitoring 13
11.3 ISS axes 18

12 Code 1 : Forecasting ISS passes **22**
12.1 Introduction 22
12.2 Julian dates improvement 24
12.3 Orbits calculation improvement 29
12.4 Time analysis 34
12.5 Focusing prediction improvement 35
12.6 Send the results by e-mail 46
12.7 Verification 47

13 Code 2: ISS FOV calculation **49**
13.1 Introduction 49
13.2 HDF files 49
13.3 Coordinate systems 51
13.4 Grid definition 58
13.5 Geometries simulation 59
13.6 Getting the final coordinates in latitude and longitude degrees 63
13.7 Verification 65

14 Georeference of lightning flashes **68**
14.1 Retrieving MMIA camera frames 68
14.2 Extract the lightning information 71

15 Conclusions **78**

16 Next steps **81**

17 Bibliography **85**

1 List of figures

List of Figures

1 Photocopy of the statement of purpose authority and responsibility . . . 2
2 ASIM modulus [1] . . . 7
3 Example of a LMA data measurement representation [5] . . . 9
4 MMIA instrument [6] . . . 11
5 Average revisit time of the ISS [7] . . . 12
6 Orbit parameters (a) [9] . . . 17
7 Orbit parameters (b) [9] . . . 17
8 ISS coordinate system [8] . . . 18
9 Pitch, yaw and roll movements . . . 19
10 Euler rotation angles [11] . . . 20
11 ISS rotation angles and axes [8] . . . 21
12 Flow chart 1 . . . 27
13 Flow chart 2 . . . 28
14 ISS tracker capture [17] . . . 33
15 ISS FOV calculation (Adapted from [18]) . . . 37
16 Flow chart 3 . . . 41
17 Flow chart 4 . . . 45
18 Flow chart 5 . . . 50
19 CTRS system [22] . . . 52
20 Example of rotating the FOV cone VS rotating the Earth . . . 53
21 ISS and Earth systems (Adapted from [23]) . . . 54
22 Summary of the coordinate system process (Adapted from [24]) 56
23 Transformation matrix calculation . . . 57
24 Discretization grid . . . 58
25 Displacement of the Earth from the top for a pitch = 5°, roll = 5° and yaw = 4° . . . 59
26 Earth simulation schema . . . 60
27 Simulation of the geometries . . . 61
28 FOV contour intersection, top and bottom parts, seen from the top of the Z axis for a non pitch, yaw and roll case . . . 62
29 Save the projection of the Nadir (Own figure) . . . 65
30 Results of the tests . . . 66

31 Output 1: MXGS detection 69
32 Output 2: Photometer analysis 69
33 Output 3: MMIA detection 70
34 Output 4: MMIA detection for oxygen and UV channels (zoom version) 70
35 The representation of the flashes by its group and radiance for case 2019/05/5 06:52:25 72
36 MMIA camera FOV example (Own figure) 73
37 Parallax definition (Own figure) 74
38 MMIA sensor FOV example 75
39 Georeference done for the case 2019/05/05 06:51:47 77
40 Actual scale of colors 82
41 Actual scale of colors vs the one proposed 83

2 List of tables

List of Tables

1 LIS parameters and performance criteria [3] 10
2 TLE technical orbital characteristics (line 1) 14
3 TLE technical orbital characteristics (line 2) 14
4 TLE technical orbital characteristics (line 3) 15
5 Results comparison 1 32
6 Results comparison 2 32
7 Results comparison 3 32
8 Days vs time comparison 34
9 Recorded step results demonstration (Version 1) 39
10 Recorded step results demonstration (Corrected version) . . . 40
11 Lightning validation data 47
12 Translation process 57
13 Corners coordinates 76

3 Acronyms

ASIM = Atmosphere-Space Interactions Monitor.

CCD = Charge-Couple-Device.

CTRS = Conventional Terrestrial Reference System.

ESA = European Space Agency.

FOV = Field-Of-View.

GNC = Guidance, Navigation, and Control software processing.

ISS = International Space Station.

LIS = Lightning Imaging Sensor.

LMA = Lightning Mapping Array.

LRG = Lightning Reserach Group.

MMIA = Modular Multi-Imaging Assembly.

TLE = Two-Line Element set.

UPC = Universitat Politècnica de Catalunya.

UTC = Coordinated Universal Time.

UV = Universidad de Valencia.

4 Abstract

First of all, it is important to say that this thesis will be developed in a research group of the UPC university, the lightning research group (LRG) and that it will be used for researching work inside the ASIM project. Which is an investigation consortium of many European universities and the European Space Agency. Being necessary to collaborate with other researchers from foreign universities.

Moreover, as the title of the thesis shows "Study on the determination of the field of view of the ASIM instruments aboard of the ISS", the thesis will use confidential and exact information from the ISS. For this reason some codes used to achieve this information, which belong to foreign researchers, will not be included in the annex. However, the use of these programs and the exact information extracted, will be commented in detail.

Speaking about the objectives, the final aim of this project is to be able to georeference the lightnings detected by the sensors of the ISS. However, as this thesis will be used during some researching campaign missions, it is important to know the exact ISS position in certain time.

This way, in order to achieve the final objectives of the thesis and use it for research work, two codes have been developed. The first one, developed in Python, it is a code that allows to know the ISS position in a certain time, which has been improved as it also allows to know the gap of time when the ISS has been or will be focusing to a desired coordinates. On the other hand, the second code is the one that predicts the exact FOV of the instruments above the ISS. Being this last one, the code used to develop a process that allows to georeference the lightnings seen by the ISS.

Finally, it is needed to say that this thesis has also been developed using also the theoretical information about the MMIA instrument's position, inclination, etc. So any changes between the theoretical values of its position and the real ones, will produce a deviation in the simulation of the georeference of the lightnings.

6 Aim

The aim of this project is to be able to georeference the lightning flashes detected by the sensors of the ISS. One of the difficulties when some analyses of the detections of ASIM (the mission that involves this project) are done, it is to know the exact position of the ISS at the moment of interest. For this reason, it is important to highlight that the first step of the project has been to develop a code that allows to know when the ISS was or will be focusing into a desired coordinates, in order to use this information for future researching campaign missions by the lightning research group.

On the other hand, the second problem when some analyses of the detections of ASIM are done, it is to know the exact FOV of the instruments above the ISS. For this reason, an initial study of the ISS attitude behavior will be done. Arriving to the development of a final code and methodology that allows to determine the field of view of the ASIM instruments aboard of the ISS and georeference the pictures of the lighnings taken by the ISS.

7 Scope

The scope of this project includes a full theoretical overview of the characteristics of the sensors above the ISS usually used to detect lightnings, focusing into LIS and MMIA. Firstly, an introduction to the ISS general attitude information and axes will be done, in order to get the knowledge for changing the coordinates values of the FOV from the ISS axes into the axes of the Earth, achieving the final latitude and longitude values thanks to the bases change.

In addition, some theoretical analysis of the orbit of the ISS will be done. Analyzing the TLE of the satellites in order to extract the enough orbit information that allows to understand the ISS attitude behavior.

Moreover, as it has been said, a first code that simulates when the ISS has been focusing in the past or will be focusing in the future to a desired point (during the next 7 days) will be done. However, it is important to highlight that to develop this program some hypothesis will be considered, getting as a result a good approximation. As it will be demonstrated.

On the other side, the second code that will be developed in this TFG, will be the one that calculates with accuracy the FOV from the instruments installed on the ISS. The extra value given to this code is that you can introduce the exact pitch, yaw and roll angles of the desired instruments from the ISS. So, in addition, it extrapolates the latitude and longitude points of the boundary FOV coordinates from any single instrument you desired.

Finally, as a conclusion, a methodology that allows to georeference the picture of the lightnings seen by the instruments from the ISS will be explained and consequently, some real cases will be analyzed. For ending this TFG, a section to improve this last step will be done.

However, it is important to highlight that as this TFG consists mainly in developing codes for helping in research investigations, non economical and environmental studies will be done deeply. Although a budget will be included as an annex.

8 Requirements

As the work developed for this TFG will be used for experimental research missions in Colombia, some indisputable requirements were defined. The main one, is to develop all the codes and processes in softwares that could be used by the LRG researchers anyway. And consequently, that can be easily sent to the different European researchers without any problem.

After that, as the first code will be also used for predicting when the ISS is focusing to a desired coordinate, the accuracy in time of the first code should be very high. It is important to highlight that the ISS is moving around 7,5 km/s, being its velocity the main problem for that.

For the second part of the project, the one that involves georeference the lightnings / pictures, the codes, process and methodologies followed, should adapt themselves to any instruments above the ISS. However, it has to be able to georeference lightnings detected by the sensors in the Earth. This should be done, in order to be able to georeference an event from any sensor and compare between them, in order to extract any useful information.

However, it is important to highlight that this last part, will be only used to georeference pictures, extracted by the MMIA sensor above the ISS. And that the methodology and codes developed will be tested with the LIS data (because it is considered the most accurate).

It will be a real challenge to achieve all the requirements. For this reason, alternative proposals will be defined along the project.

9 Justification

The LRG has been working on the ASIM mission project since 2007, preparing the ground facilities for research activity produced by the lightnings, and analyzing the TLE (Transient Luminous Events) detected by the ISS.

In fact, the exact objective of the LRG participation in the ASIM project, it is to study new processes and techniques that will be used in the aerospace pitch, for reducing the amount of lightnings that impact to the airplanes. Reducing, this way, the damages produced and consequently, the costs.

This way, for advancing with the ASIM mission project, it is needed to recollect all the possible lightning information data. For this reason, it is highly important to be able to monitoring the ISS at the experimental research missions in Colombia, in order to connect all the instruments/sensors in the Earth for collecting all the possible information.

Moreover, the actual status does not allow to compare with high accuracy, the different information of the lightnings seen by different instruments. For this reason, in order to know the similarities and differences (understanding the possible errors) between them, the possibility to georeference the lightnings could be very useful. In order to extract, for example, if a lightning seen by the MMIA sensor it is or not seen by another one. Determining mistakes in the attitude of every single instrument, for example.

These are some of the reasons that justified the development of this TFG. However, as the investigation will be still done, new justifications, needs and requirements can appear.

10 State of the art: ASIM and ISS-LIS

10.1 ASIM introduction

As it is commonly known, the ISS has been harbouring several experimental instruments for different research missions. One of them, is the Atmosphere-Space Interactions Monitor, called ASIM.

ASIM is a climate observatory for the ISS launched on 02/04/2018, developed by the ASIM consortium for ESA. The ASIM consortium is formed by Terma A/S, Technical University of Denmark, University of Bergen, University of Valencia, Polish Academy of Sciences Space Research Center, and OHB from Italy. ASIM is expected to give new insights into climate processes that can improve climate models by quantifying the effect of electrical and chemical processes at the space-atmosphere boundary.

The ASIM payload has a mass of 314 kg (692 lb) and consists of two subsystems, the CEPA (Columbus External Payload Adapter) and the DHPU (Data Handling and Power Unit), which form the structure and electrical connections, respectively, of the Columbus module. Moreover, the ASIM payload also includes two scientific instruments called MMIA and MXGS (Modular X and Gamma Ray Instrument). MMIA will be explained in detail in section 9.3, however, just say that the objective of MMIA and MXGS is to form a climate observatory designed for detecting transient luminous event and terrestrial gamma ray flashes [6]. Here, a representation of the ASIM modulus is shown below :

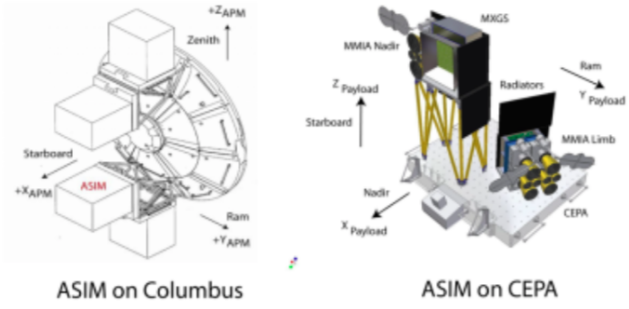

Figure 2: ASIM modulus [1]

10.2 LIS

LIS is a space-based lightning sensor aboard the ISS, that records the time of occurrence of a lightning event, measures the radiant energy and estimates the location during both day and night conditions with high detection efficiency. For this reason, its ata sets are used to detect the distribution and variability of the total lightning occurring in the Earth's tropical and subtropical regions. Moreover, as it is said in the data user guide of LIS-ISS [2], the LIS instrument also provides periodic background images that help with the navigation and calibration monitoring of the ISS.

Also explain that the use of the ISS as the monitoring satellite for the LIS sensor, provokes that only the 81 % of the Earth's surface can be analyzed, but includes 98 % of the global lightning on an annual basis. As it can be noticed in the graph done by NASA posted in "Sharing earth observation resources website"[3] or figure 1.

As it has been said in the State of the art chapter, the ISS Lightning Imaging Sensor is a space-based lightning sensor that records the time of occurrence of a lightning event, measuring its radiant energy and estimating the location. Allowing to map (space and time) individual flashes with thousands of detections. Also, it is important to comment that ISS-LIS data is being exported in HDF-4 format, which consist of near-real time scientific data and background data that it is still being checked. Being only available since 2017, provoking to analyze mainly the lightnings of 2018.

It is important to comment that for validating the data achieved from the ISS-LIS, the LRG researchers use LMA sensors data for comparing the results. The LMA is a network of time-of-arrival sensors that passively receive very high frequencies impulses from electrical breakdown within thunderstorms. Determining, this way, the exact locations of in-cloud lightning. Right now, the LIS gives a very high accuracy, defining the lightnings position with exactitude.

Right now, the LRG has the control of two LMA sensor area [4], one placed in Colombia and the other in Delta del Ebre. Here, an example of the LMA data of Delta del Ebre is presented:

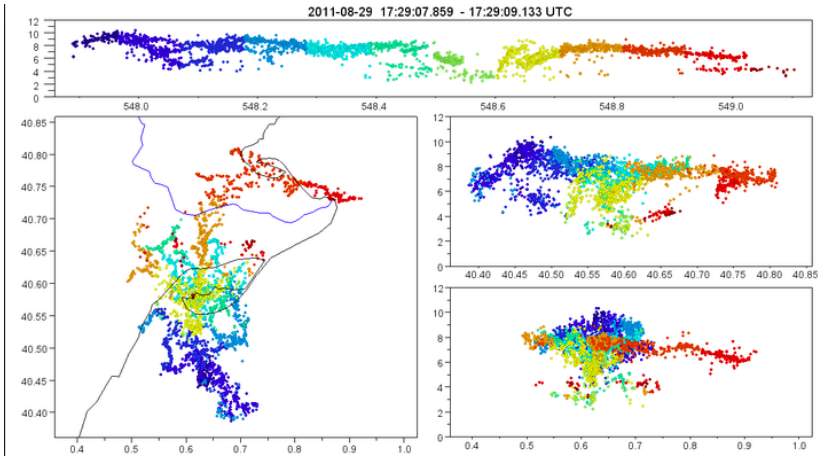

Figure 3: Example of a LMA data measurement representation [5]

In this figure, the development of a lightning during a period of time can be seen. Precisely, the picture from the top shows the height of the lightning vs the time, while one from the left (at the bottom) shows the coordinates in latitude and longitude of the lightning, being able to see carefully the zone when it could be seen. Finally, the two pictures from the right, show the height of the lightning vs the latitude (top one) and longitude (bottom one).

Speaking about the sensor, the design sensor employs an expanded optics wide FOV lens, combined with a narrow-band interference filter, that focuses the image on a small, high-speed 128 x 128 CCD focal plane. As it can be seen in the LIS technical characteristics table showed in table 1. The signal is read out from the focal plane at 500 images per second into a real-time event processor for event detection and data compression. The resulting lightning data is formatted, queued, and sent to the spacecraft for transmission to ground stations.

Here are presented all the technical specifications of LIS sensor :

Parameter	Value
FOV (Field of View)	80° x 80°
Pixel IFOV (Instantaneous FOV)	4 km
Interference filter Wavelength	777,4 nm
Interference filter Bandwidth	1 nm
CCD array size	128 x 128 pixels
Dynamic range	over 100
Detection efficiency	90 %
False event rate	bellow 5 %
Measurement accuracy location	1 pixel
Measurement accuracy intensity	10 %
Dimensions sensor head assembly	20 x 37 cm
Dimensions electronic box	31 x 22 x 27 cm
Instrument mass	25 kg
Instrument power	35 W
SNR (Signal to Noise Ratio)	6
Telemetry data rate	8 kbit/s
Telemetry format	PCM (Pulse Code Modulation)

Table 1: LIS parameters and performance criteria [3]

10.3 MMIA

ASIM scientific instruments include 6 cameras, 6 photometers and one X and gamma-ray detector. Mostly, the cameras and photometers are directed forward towards the horizon (ram, limb). However, the two cameras, two photometers and the X and gamma-ray detectors used in this TFG, are directed downwards at the Nadir direction.

The cameras and the photometers constitute the MMIA. The MMIA instruments observe in different optical spectral bands. ASIM uses optical observiations in carefully selected bands in order to filter out data with TLEs from the lightning data. Since downlink is limited, those algorithms are implemented in the on-board software. The ASIM MMIA instrument is capable of observing 12 frames per second continuously in the 777.4 nm and 337 nm bands, both only 5 nm wide. Combined with 100 kHz photometer data from the same two bands in addition to a 180-230 nm band, data is filtered en real time to optimize the available downlink capability allocated to ASIM on ISS [6].

One of the requirements of this TFG, is to develop a process or methodology that allows to georeference the MMIA pictures of the lightnings extracted from the code done by the UV researchers. In order to do this, the pictures from the 777 band (the oxygen one) will be the chosen and the data will be compared with the one given by LIS, as LIS gives highly accurate data. However, this information will be exposed deeply later, in chapter 13. Here a picture of MMIA instrument is presented :

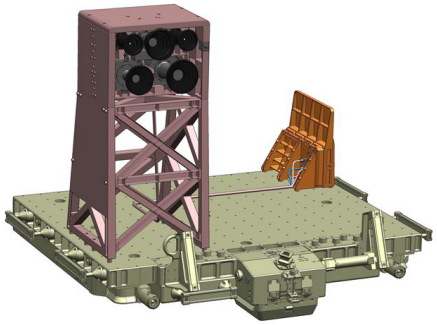

Figure 4: MMIA instrument [6]

11 ISS characteristics

11.1 ISS orbit information

The ISS has a nearly circular orbit inclined at 51.63° to the equator with an average altitude between 330 and 435 km. However, during the labour time of this TFG the altitude of the ISS has been always between 409 and 415 km. Comment that the ISS moves along its orbit at a velocity of approximately 7,6 km/s, orbiting the Earth roughly 92 minutes and completing 15.5 orbits per day, having in Europe and South America (the places were the LRG has sensors) a revisit period below 3 days. As it can be seen from this picture extracted of the research article [7].

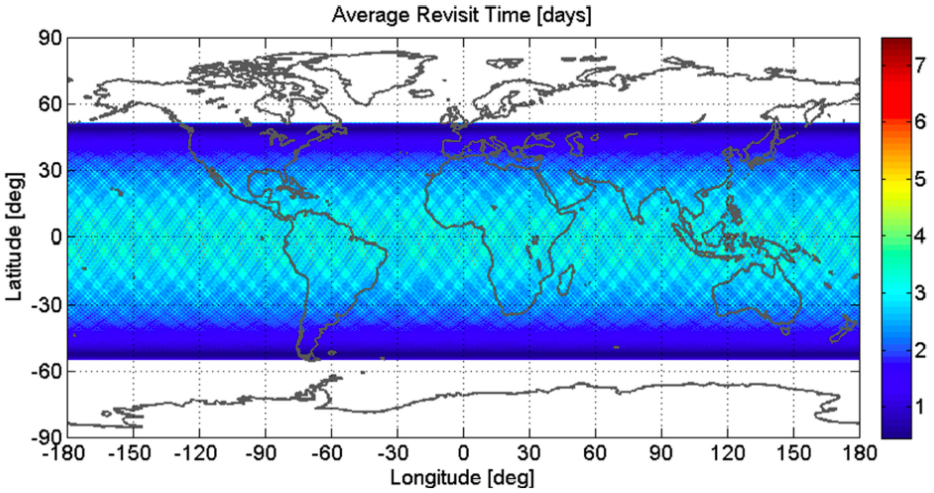

Figure 5: Average revisit time of the ISS [7]

Comment that the ISS altitude decreases between 100 or 200 meters per day, caused by the atmospheric drag. However, the rate of descent is not constant, because of the changes in the density of the outer atmosphere, which is a consequence of solar activity. Hence, some periodic re-boosts are needed to counteract this decrease. Highlight that the re-boosts can take place at intervals anywhere between 10 and 80 day, as it is said in this ESA European Users Guide to Low Gravity Platforms, chapter 7 [8].

11.2 Orbit monitoring

For control and monitoring the ISS, few parameters should be previously known. For this reason, in order to make things simple, NORAD (North American Aerospace Defense Command) and NASA produced some perturbations models used to calculate orbital state vectors of satellites and space debris relative to the Earth-centered inertial coordinate system. As reproducing every single time the equations of the desired model could be extensive, a kind of an orbit attitude identification code (called TLE) was created.

TLE is a Two Lines Element data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time. This way, TLE gives the enough information that can be used for suitable predict the position and velocity of the satellite at any date in the near past or future. According to that, it is important to highlight that TLEs can describe the trajectories only of Earth-orbiting objects.

In the case of the ISS, even though 3 or 4 TLEs are created every single day, when an historical satellite position prediction is being done, simulations show that there is not an important change using the first TLE of the correspondent day or the others. In fact, as it was done in the first code programmed in this TFG (the one that determines when the ISS was or will be focusing into a desired coordinates), you can predict 8 days in advanced considering only the TLE of the actual day.

The TLE data representation is specific to the simplified perturbations models produced by NORAD and NASA (which are SGP, SGP4, SDP4, SGP8 and SGP8), so any algorithm using a TLE as a data source must implement one of the SGP models. In the case of the first code programmed in this TFG, the Python function "satellite compute" of the library "Ephem" is used to calculate the position at each time of the ISS. Achieving perfect results.

In order to analyze the different information that can be extracted from a TLE, an example of TLE from the ISS and its information, is presented below :

- Example :
  ISS (ZARYA)
  1 25544U 98067A 08264.51782528 -.00002182 00000-0 -11606-4 0 2927
  2 25544 51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537

Parameter	Value
Satellite name	ISS (ZARYA)

Table 2: TLE technical orbital characteristics (line 1)

Parameter	Value
Line number	1
Satellite number	25544
Classification (U=Unclassified)	U
International Designator (Last two digits of launch year)	98
International Designator (Launch number of the year)	067
International Designator (piece of the launch)	A
Epoch Year (last two digits of year)	08
Epoch (day of the year and fractional portion of the day)	264.51782528
First Time Derivative of the Mean Motion divided by two	.00002182
Second Time Derivative of Mean Motion divided by six (decimal point assumed)	00000-0
BSTAR drag term (decimal point assumed)	-11606-4
The number 0	0
Element set number	292
Checksum (to verify data integrity)	7

Table 3: TLE technical orbital characteristics (line 2)

Parameter	Value
Line number	2
Satellite number	25544
Inclination	51,6416°
Right ascension of the ascending node (degrees)	247.4627
Eccentricity (decimal point assumed)	0006703
Argument of perigee (degrees)	130.5360
Mean Anomaly (degrees)	325.0288
Mean Motion (revolutions per day)	15.72125391
Revolution number at epoch (revolutions)	56353
Checksum (to verify data integrity)	7

Table 4: TLE technical orbital characteristics (line 3)

- Definitions :
  BSTAR drag term : BSTAR is a way of modelling aerodynamic drag on a satellite in the SGP4 satellite orbit propagation model.

  Checksum : A checksum is a small-sized datum derived from a block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage. It is usually applied to an installation file after it is received from the download server. By themselves, checksums are often used to verify data integrity but are not relied upon to verify data authenticity.

  Inclination : Orbital inclination measures the tilt of an object's orbit around a celestial body. It is expressed as the angle between a reference plane and the orbital plane or axis of direction of the orbiting object.

  Right ascension of the ascending node : The ascending node is the point where the orbit of the object passes through the plane of reference. Depending on the orbit, it could be defined as :

  1. For a geocentric orbit, equatorial plane is defined as the reference plane and the First Point of Aries is defined as the longitude's origin. In this case, the longitude is also the right ascension of the ascending node.

2. For an heliocentric orbit, the ecliptic is defined as the reference plane, and the First Point of Aries is defined as the origin of longitude. The angle is measured counter-clockwise (as seen from north of the ecliptic) from the First Point of Aries to the node.

3. For an orbit outside the Solar System, the plane tangent to the celestial sphere at the point of interest (called the plane of the sky) is defined as the reference plane, and the perpendicular projection of the direction from the observer to the North Celestial Pole onto the plane of the sky, is defined as the origin of longitude. The angle is measured eastwards (or, as seen by the observer, counter-clockwise) from north to the node.

Eccentricity : The orbital eccentricity of an astronomical object is a parameter that determines the amount by which an orbit deviates from a perfect circle. A value of 0 is a circular orbit, values between 0 and 1 form an elliptic orbit, 1 is a parabolic escape orbit and greater than 1 is a hyperbola.

Argument of perigee : Is the angle from the body's ascending node to its periapsis, measured in the direction of motion.

Mean Anomaly : Is the fraction of an elliptical orbit period that has elapsed since the orbiting body passed periapsis, expressed as an angle which can be used in calculating the position of that body in the classical two-body problem.

Mean motion : Is the angular speed required for a body to complete one orbit.

Some pictures showing the characteristics and parameters defined above, are included:

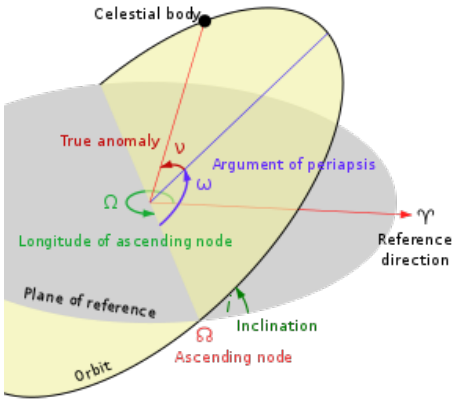

Figure 6: Orbit parameters (a) [9]

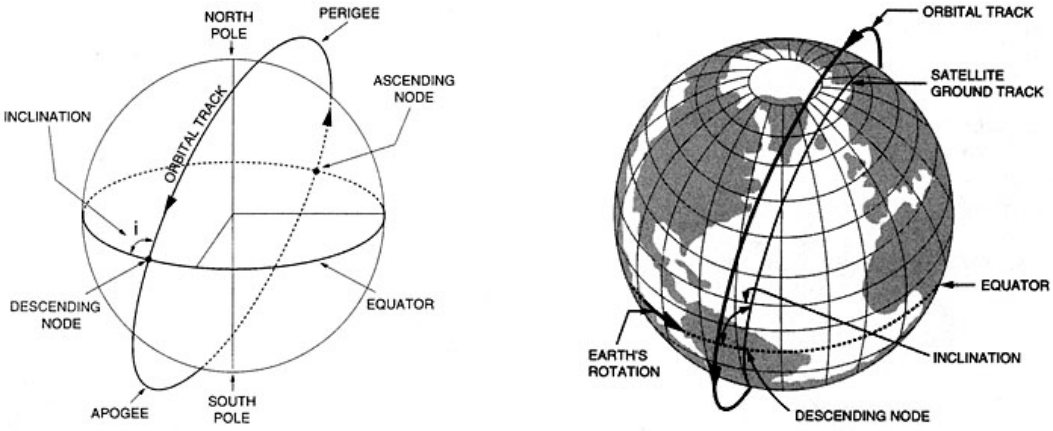

Figure 7: Orbit parameters (b) [9]

11.3 ISS axes

The position and velocity extracted from the ISS, is saved in the HDF files that have their coordinate system in the center of the Earth. However, as the final objective is to predict the FOV of the instruments above the satellite (as if we were taking a photo), it is much easier to be positioned above the ISS. Consequently, using the ISS axes. For this reason, as the result should be expressed in latitude and longitude coordinates, it will be needed to transform the resulting coordinates into the system used in the Earth.

This way, following the ESA guide [8], it can be seen that many coordinates axes can be used for the ISS. Even thought for the code developed in this TFG, the ISS coordinate system used has been the one that assumes the ISS velocity vector as the X vector. While the Nadir vector (the vector that connects the satellite with the center of the Earth) is used as Z vector. And consequently, the Y axis is the one achieved doing the cross product between the Z and X vector. Highlight, that in order to avoid general problems, all the operations that include the different coordinate systems, will be carried with their correspondent unitary vectors.

According to the above information, the following ISS axes system is achieved:

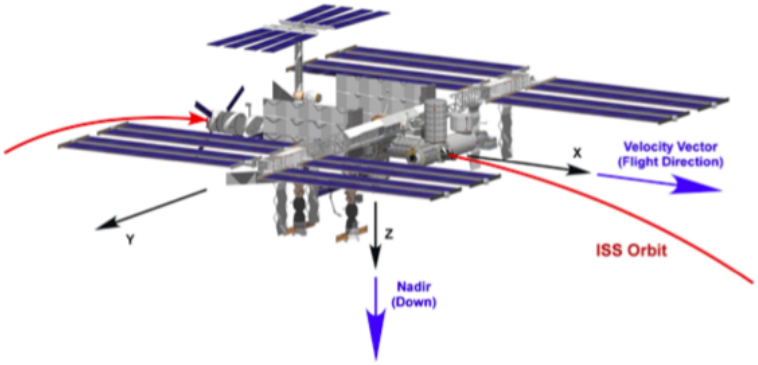

Figure 8: ISS coordinate system [8]

It is important to highlight that satellites do not keep always the same position, being free to rotate from their main axes and consequently, deviating from their some degrees. Producing, as a result, a new coordinate system.

This way the three possible movements are :

Yaw : Movement that produce the nose moving left or right. It is a rotation about the Z axis (the Nadir) and measures the degrees between the first X axis and the new one.

Pitch : Movement that produce the nose moving up or down. It is a rotation about the Y axis and measures the degrees between the horizontal plane and the new X vector.

Roll : Rotation about an axis running from nose to tail, the X axis. It measures the degrees between the horizontal plane and the new Y vector.

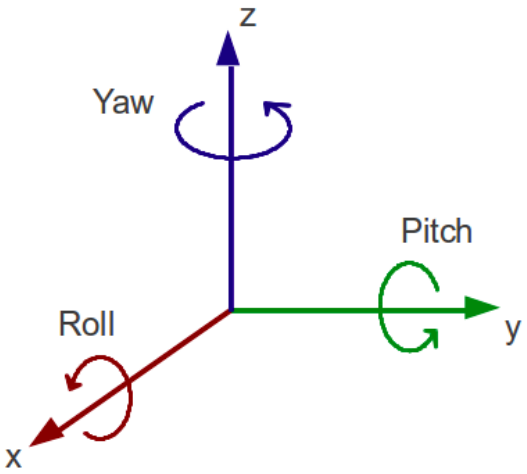

Figure 9: Pitch, yaw and roll movements

So because of the pitch, yaw, and roll movements, the camera sensor will not be focusing into the Nadir direction. Being necessary to use a transformation matrix between the ideal ISS axes and the real ones, the ones modified by the pitch, yaw and roll degrees movements. This transformation matrix is called Euler rotation matrix and was introduced by Leonhard Euler to describe the orientation of a rigid body with respect to a fixed coordinate system.

In order to define this matrix, the pitch, yaw and roll degrees of the ISS should be known in advanced. Obtaining them from certain programs developed by researchers of the UV (Universidad de Valencia) working in the ASIM project. However, if the readers of this TFG can not get into this accurate information, this website can be used for getting similar information for further studies [10].

Highlight that in case of the ISS, a change of 1° in roll, can provoked a 10 km deviation from the Nadir ideal position. For this reason, it is important to know exactly this data. This way, the next rotation angles are defined:

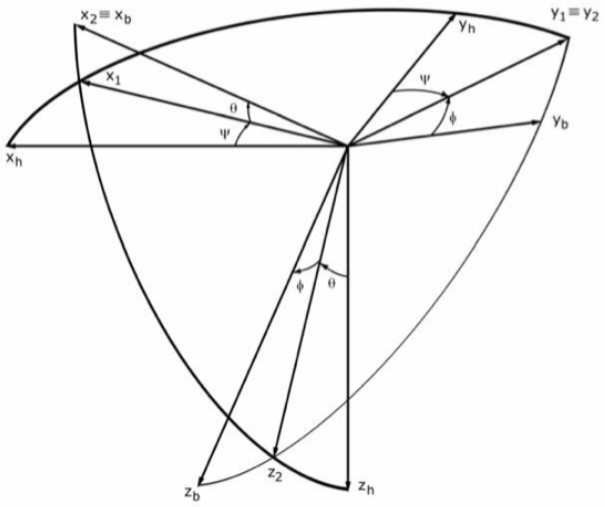

Figure 10: Euler rotation angles [11]

Understanding : Yaw = ψ, Pitch = θ, Roll = ϕ.

The next picture, show the ISS axes with their correspondent rotation axes definition. Comment, that the picture is added in order to present to the reader the association between pitch, yaw and roll and the correspondent axis in the case of the ISS.

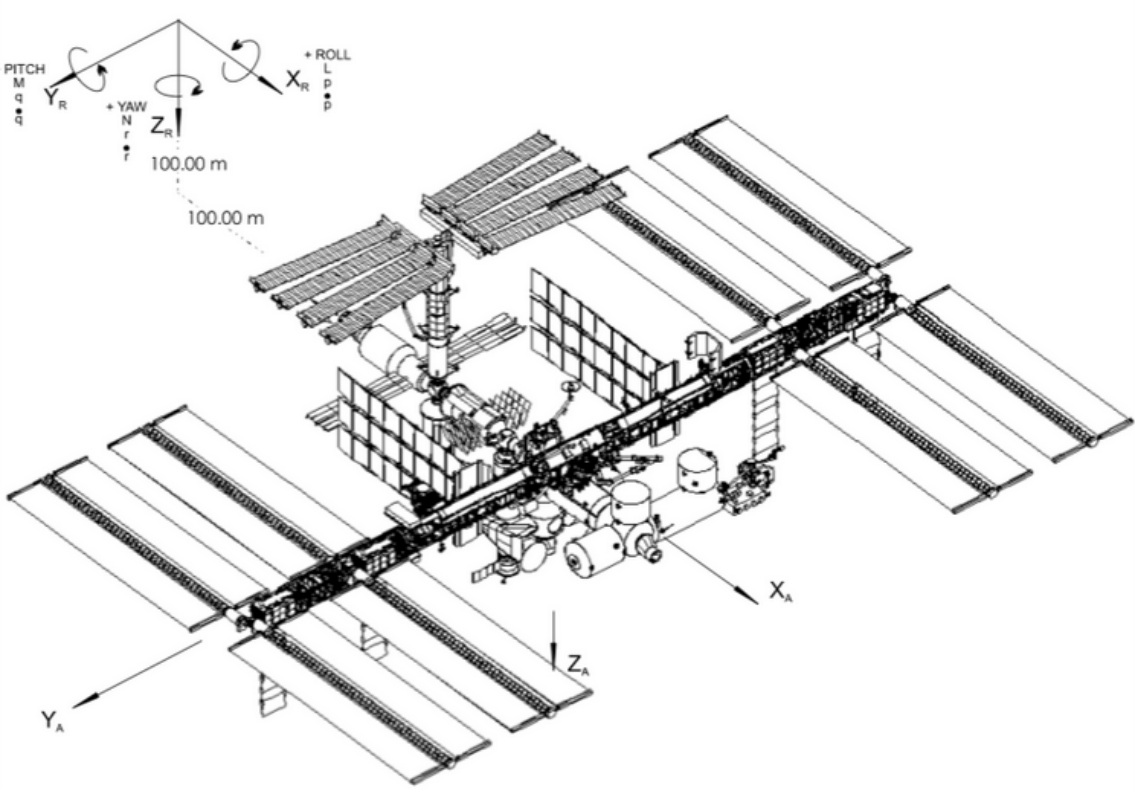

Figure 11: ISS rotation angles and axes [8]

This way, the rotation matrix that allows to pass from the ideal position to the real one, is achieved :

$$\begin{pmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi) & sin(\phi)cos(\theta) \\ cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi) & cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi) & cos(\phi)cos(\theta) \end{pmatrix}$$

12 Code 1 : Forecasting ISS passes

12.1 Introduction

One of the requirements of this TFG was to create a code able to predict the position of the ISS in certain future and past time, in order to to know the gap of time when the ISS will be passing over a desired position while some researching campaign missions are being done. It is important to say, that this starting idea has been improved, developing a coda able to determine when the ISS is focusing to a desired area/point. Comment that the code explained in this section has been done in Python language, which I have learned to code during this process.

In order to have a starting point, one of the LRG researchers gave me an erroneous code that pretended to predict when the ISS will be passing over Australia during the next 5 days. So, at this chapter, the researcher's code will be explained, as well as also the errors it contains. Afterwards, the different chosen solutions for the problems will be explained, doing emphasis at the obligatory requisites that the final solution need to fulfill. Finally, the code developed during this TFG and correspondent algorithms will be explained in detail.

After analyzing the code developed by the LRG researcher, the mistakes found were :

- 1. It is not able to do an historical prediction (as it can only calculate 5 days in advanced) and the code bugs when it changes of month or year during the simulation.

- 2. The code does not calculate the 16 orbits that the ISS does. The top and bottom ones, are not taken into account. So some useful data is erased.

- 3. The code only works for areas that have the same size as Australia. Does not work for Switzerland or smaller areas, for example, so it is a big problem for areas of a size as the LMA sensor area.

- 4. The code only shows when the ISS passes over a selected area. It does not calculate when it is focusing to a desired area.

Once that all the problems of the researcher's code have been exposed, the solutions proposed and chosen will be explained in the following sections. Highlight that in order to make a better comprehension for the reader, the parts of the code programmed in this TFG and its different solutions for the original problems, will be explained individually. For more information, all the codes developed, are able in the annex of the TFG.

First of all comment that the needed libraries used, are:

Datetime: Allows to operate and work with different date formats.

Ephem: Allows to calculate/predict the different ISS position. It is a library used to work with TLE of different satellites.

Smtplib: Allows to be able to connect Python with an email. In order to be able to send automatically the results to other emails of different researchers.

The following sub-libraries are imported to be able to send the results by email.

```
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders
```

Math: To be able to work with degrees, radiants and make different mathematical calculations.

12.2 Julian dates improvement

Once the code developed by the researcher has been checked, the first observed problem was the impossibility to work with historical data, which is a big problem. The fact of not being able to know the past passes, provokes that you are not being able to review when the ISS has been above the colombian LMA sensors area during the 2018 year. Avoiding to extract the gaps when the ISS was over the desired are and avoiding to compare the LMA data with the LIS one too Which is important for getting deeper lightning detailed information and analysis between the sensors placed in the Earth and in the ISS.

Mainly this problem happened because the researcher's program worked with a Gregorian calendar [12], a calendar that uses a year/month/day format adding +1 to the actual day that has been simulated. Disabling the execution of the program when a month or a year changes. Basically, the use of this type of calendar provokes that as the simulation is increased day by day, it can happen for example, that suddenly the code searches the TLE of the day 32 of October, that does not exist. Getting a bug as a consequence.

For this reason, it was necessary to change the methodology process into one that uses a Julian Date calendar, a calendar made by a continuous count of days since the beginning of the Julian Period [13], used primarily in softwares that need to calculate elapsed days between two events, without any problems.

So as it has been previously said at section 11.2, in order to calculate the ISS position the program needs to know the TLE of the satellite for the desired day. Just remember that the TLE of a satellite is the code that includes all the orbit information. For this reason, the program of the researcher connected the Python code into a website called Celestrack [14] (that gives the actual day TLE of many different satellites) in order to download the TLE of the ISS and use it to predict the position of the ISS during the following 5 days. Also remember that with a single TLE, you can predict a week in advance. However, highlight that the requirement of this TFG is to develop a program able to calculate the historical position of the ISS too.

For this reason, a file with all the TLEs since 1/1/2005 was created. Highlight that this data base file was created as a CSV in order to improve the speed of execution of the code, because reading a CSV file in Python is faster than reading a TXT. Moreover, an algorithm that searches the desired TLE of the day in the CSV historical list and saves it as a Python variable, was developed. Consequently, using an historical TLE list and a Julian calendar, allowed to simulate all the several wanted days without the problem of getting bug when a month or a year changes during the simulation. Improving the initial code developed by the researcher.

This way, the working methodology of the algorithm that searches the desired TLE is the following :

- 1. First of all it is necessary to introduce as inputs the initial and final dates of the simulation, in order to execute the code as a daily loop. For making the code simple to use, the inputs days should be introduced in Gregorian format (year/month/day) being later converted into Julian format.

- 2. After that, as it shown in section 11.2, in order to search the exact TLE of the desired day, the last two digits of the year and the numerical day of the year should be extracted.

  For this reason, an algorithm able to read the year of the desired day and extract its last two digits, was made. Once the year of the desired day is extracted, in order to achieve the numerical day of the year, the difference plus one between the simulation day and the first of January of the extracted year should be done in Julian date format. Getting as a result, the exact numerical number for the simulation day and achieving all the necessary data to search the TLE.

  Example: 30/5/2019 is day 151 of the year.
  Julian date of 30/5/2019 = 2458634
  Julian date of 01/01/2019 = 2458485
  Numerical date = 2458634 - 2458485 + 1 = 151

Finally, the year and the numerical day are saved as variables. However, highlight that as the program is working with integer numbers, it is necessary to append a zero for the numerical day in case the number is smaller than 10, or two zeros if the numerical number of the day is between 10 and 99. Moreover, as it is seen in section 11.2, the final point for achieving the TLE day information data, is also included.

Following, an example of a TLE searching variable is presented :

Example of searching variable:
19 of February of 2019 : **variable = 19049.** (Last two digits of the year = 19 and numerical day = 049).

Example TLE:
ISS (ZARYA)
1 25544U 98067A **19049.**59717910 .00001324 00000-0 28051-4 0 9992
2 25544 51.6402 230.9169 0005853 42.7111 353.1664 15.53275152156834

- 3. Once that all the necessary data to search the TLE is achieved, an algorithm able to search line by line and word by word at the historic CSV list is made. Basically, the algorithm searches in the CSV list the line that has the value "19049.", getting as a result the first line of the desired TLE. So after saving the whole line, in order to have both TLE lines, the next line is saved too.

As the name of the satellite would not change, it can be included as a variable in the program, so it is not necessary to search it. Attended that the required precision is very high and that the computational loss of time searching and executing the TLE daily is minimal, however a single TLE allows to predict 7 days in advanced, in order to obtain an optimal precision, the developed program will use the TLE of each day.

The parts of the code explained above, are exposed in the following flow charts. Also the correspondent code parts of this section are included in annex 1, before the full program exposition.

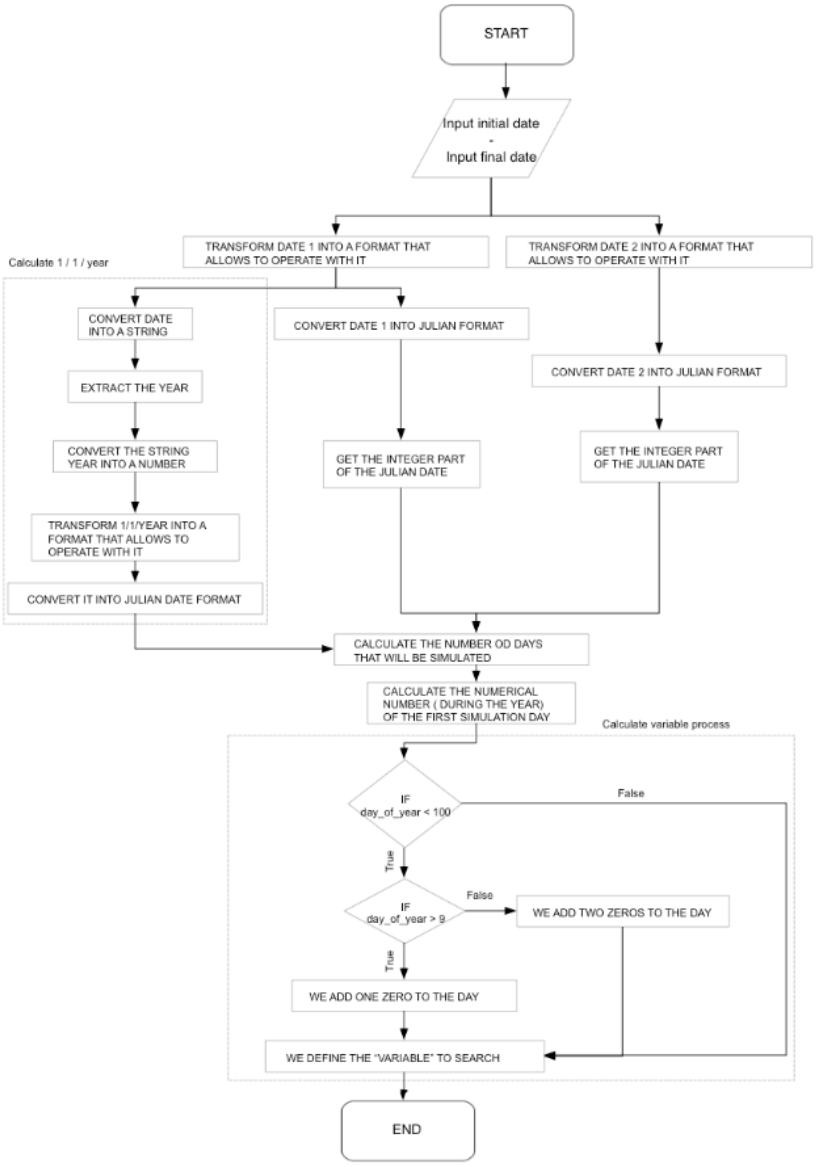

Figure 12: Flow chart 1

Extract the variable to search the TLE flow chart

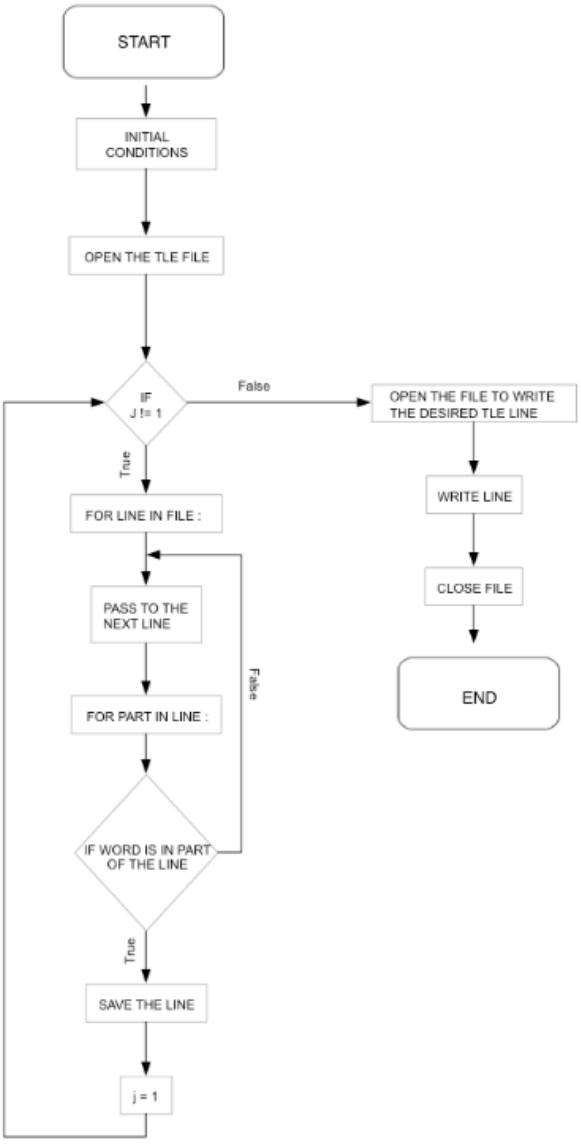

Figure 13: Flow chart 2

Search the TLE in the CSV list flow chart

12.3 Orbits calculation improvement

Once the TLE of the day is obtained, the following step of the program is to compute/predict the position of the ISS during the whole day.

Remember that when the researcher's code has been checked, it has been found that it is not able to calculate all the orbits that the ISS does. As it does not calculate the 2 boundary orbits, the top and bottom ones, that are not able to being simulated. Provoking to be the program erroneous and therefore, no valid in this aspect. Obviously if the final objective is to use the code to compare LMA and LIS data, it is not acceptable that two orbits are not defined. For this reason, as the program needs a very high precision, all orbits must be calculated/predicted as a requirement.

First of all, comment that the library used to determine the ISS position is the Ephem one that it is commonly used in Python for performing high-precision astronomy computations. For achieving the satellite position, the TLE has to be adapted to the "Ephem" library format, using the "readtle" function. Afterwards, the function "compute" is also used to get the latitude and longitude of the desired satellite in a specific time.

This way, the work methodology of the algorithm that calculates the position of the ISS is the following :

- 1. Write the TLE in a format able to be read by the "compute" function and save it as a variable called "sat". It is necessary to use the "readtle" function too.

- 2. Remember that this code works with Julian dates in order to avoid problems when a year or a month is changed. For this reason, as the "compute" function needs an exact time in Gregorian format with hours, minutes and seconds included in order to calculate the position of the ISS, the "datetime" function is used to make the 5 seconds jump. However the change of day is done with the Julian format.

- 3. The function "compute" calculates the position of the satellite given a time of the iteration and a TLE. Highlight that the daily loop starts at time 00:00:01 of the desired day and finishes at 23:59:59.

- 4. The "compute" function has a module able to express the latitude and longitude of the satellite. However, in order to work and compare this data, it is necessary to express the data in degrees. This way, the function "ephem.degrees" is used.

The corespondent part of the code, that uses the libraries and functions to calculate the position of the ISS given a TLE is placed in the annex.

So, why the researcher's code was not able to compute/calculate all the orbits of the ISS ? Analyzing it in detail, it can be observed that the code of the researcher, uses the function "newton", which pretends to use the Newton–Raphson method to simulate and extrapolate the position of the ISS between two periodic iteration jumps. In order to know how when the ISS is crossing over the desired area he has selected. Comment that in numerical analysis, Newton's method is a method used for finding successively better approximations to the roots (or zeroes) of a real-valued function. It is one example of a root-finding algorithm [15].

Surely, the orbit mistakes produced in the researcher's code were provoked by the use of the "newton" Python function. This function, produced mistakes, for example, while the ISS is passing over the 0 degrees (because of the change between the negative degrees value and the positive ones). For this reason, it is not strange that this function could produce errors while it was extrapolating the position of the ISS between two steps.

As a solution for this problem, attended that the Ephem function does not produce mistakes calculating the satellite position, that the required precision is very high and that the computational loss of time computing/predicting the ISS position was minimal, the best option considered and therefore chosen was to calculate the ISS position using the "compute" function every time a single jump was done. Making time jumps about 5 seconds.

Improving, this way, the precision of the results respect the researcher's program (which makes the jumps in a scale of 2 minutes) and being able to compute and predict all the 16 orbits. Deleting the before problem as a result of non using the newton's library and calculating the exact ISS position at every step. Just as a comment, if it is considered by the reader, the LRG researcher obtained some useful information to develop his own code from the website [16].

It is important to highlight, that the speed of the program was not affected at all. As the program developed in this TFG is able to simulate the position of the ISS during 365 days (making 5 seconds jumps) in a time below three minutes and a half. However, if the iteration gap time is increased (using jumps about 15 seconds), the computational time is reduced too. Being able to simulate 486 days in less than 3 minutes, for this case.

Although, as the ISS is moving around 7,6 km/s and the requirement of getting a good precision is desired, the gap time chosen for the final code was about 5 seconds. Point that all the time data is processed and worked with an UTC format, the primary time standard by which the world regulates clocks and time.

For the verification and validation of the results of the program, three methods were considered in order to check the ISS Nadir position :

1. **Use the historical ISS tracker website** [17]: This website allows to know the latitude and longitude coordinates of the ISS in a certain exact time.

2. **The HDF files**: This files were received from other UV ans ESA researchers. However they can include a very small error.

This way it is achieved :

For the day 2018-11-28 and Time (UTC) = 19:21:53 the Nadir is :

Criterion	Latitude(°)	longitude(°)
HDF ESA researchers	-4,6	103,96
ISS Tracker	-4,7	103,95
Developed code	-4,67	103,95

Table 5: Results comparison 1

For the day 2018-11-02 and Time (UTC) = 05:09:31 and the Nadir :

Criterion	Latitude(°)	longitude(°)
HDF ESA researchers	3,573	-65,09
ISS Tracker	3,58	-65,1
Developed code	3,56	-65,09

Table 6: Results comparison 2

For the day 2018-10-25 and Time (UTC) = 21:13:32 and the Nadir :

Criterion	Latitude(°)	longitude(°)
HDF ESA researchers	101,55	8,42
ISS Tracker	101,52	8,43
Developed code	101,53	8,38

Table 7: Results comparison 3

As the results of the the Nadirs from the HDF files sent by the ESA researchers, the ISS tracker [17] and the obtained from the developed program are almost equal, the developed algorithm can be considered as validated. Moreover, highlight that many other orbits and points were checked.

Here a capture from the ISS tracker web is included.

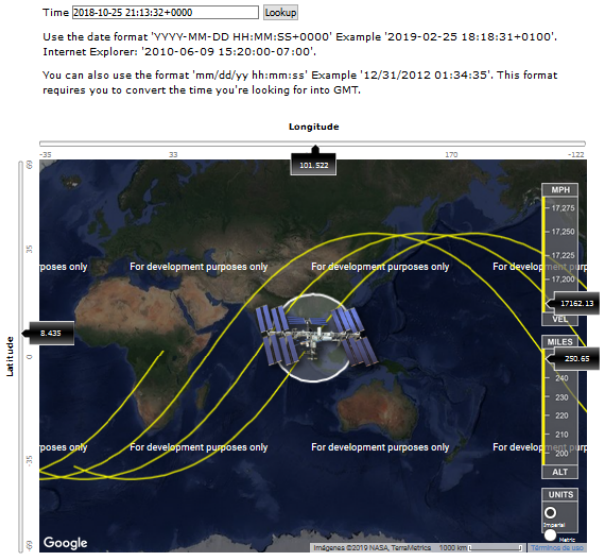

Figure 14: ISS tracker capture [17]

3. Use the lightning data able from LIS: This method will be deeply explained at the section reserved for the verification of the whole program. However, as an introduction, the methodology followed is to select a specific area contour and check if all the lightnings produced inside are also included inside the gaps of time that the program says the ISS is focusing to the desired area. This process can be done for checking this way if the program works properly or not, as the LIS sensor it is placed above the ISS. However it is important to highlight that the LIS data is actually being corrected in order to perfect it. So it is not strange if some (very few lightnings) missed.

12.4 Time analysis

The researcher's code was created for predicting in the next 5 days, when the ISS will pass over Australia. For this reason, as it has been said previously, the researcher used jumps of 2 minutes for each iteration.

What happened ? Obviously when the selected area was much smaller than the Australia size, as the ISS moves around 7,6 km/s, the ISS jumped the area between two iterations and as a consequence, the code produced a bug not working properly. It is necessary to spotlight that the bug was produced because of the Newton function, as the file jumped the whole area between steps, not detecting it. As a solution for this problem, the time between iterations was reduced to 5 seconds and therefore, it was possible to select an smaller area.

The request LMA areas for the LRG are about 1,5° of side, which is roughly 170 km. But if there is a case where an smaller area is needed, the code can adapt the precision (in order to not jumping the area) decreasing the gap time. However, the necessary time for making all the iteration will increase.

The following table, shows the days simulated vs the computation time needed for running the whole code, using 5 seconds jumps.

Days	time (s)
1	0,35
7	3,24
14	5,39
50	19,33
100	38,02
150	57,84
250	98,21

Table 8: Days vs time comparison

As it can be seen, the velocity of the code is quite good, being able to simulate 100 days in almost half a minute. Highlight that this test was done for Barranca in Colombia.

12.5 Focusing prediction improvement

The program of the researcher calculates when the ISS will pass above a square area in the next 5 days. However, instead of this, the aim of the program developed in this TFG is to predict the moment, in a recent past and future time, when the ISS was or will be focusing at some concrete coordinates.

In order to achieve this requirement, the following hypothesis were considered :

- 1. The ISS is at a fixed height of 400 km above the earth. Although, we have viewed that it can change a few.
- 2. The pitch, yaw and roll of the ISS is not considered (This hypothesis was caused by the lack of information).
- 3. It is assumed that the 40° of the LIS FOV are correct, as the technical information of the LIS sensor says. Moreover, it is also considered that there is not a variation of this value provoked by the deterioration of the sensor.
- 4. The projection screen ("the floor") is considered as a horizontal plane. As a result, the projection of the FOV resulted a circle placed 400 km under the ISS with its center in the Nadir.

Therefore, supposing that those hypothesis are valid, a circle projection about 336 km of radius is obtained. This value was achieved multiplying the altitude from the Earth by the tangent of the FOV and can be translated into a circle of 3 degrees of radius, at the case of Colombia (1° = 111 km), or about 4 degrees in the case of Europe (1°=80 km). Those two sites are considered, because they are the ones with LMA data and will be the ones checked.

Highlight that for making the code more dynamic and computational faster, two areas have been designed and used. The first one is a 336 km circle area which defines the FOV area, the area where the ISS will be focusing to the desired coordinates while it is passing over. Consecutive, a second area has been also defined. This time it is a big square area used as an approximation area. In this case, this area is defined as a square area of 8 degrees per side with the desired point in the middle. Used for refusing, with a quickly comparison, the computational jumps that are outside the area.

As the position of the ISS is being calculated every 5 seconds, it is necessary to check in each iteration if the ISS is near the desired area or not. However, comparing float numbers is much easier and faster than calculating the exact difference in km between two points, for this reason the first area is defined in degrees and it only compares the values of the latitude and longitude coordinates from the ISS, with the ones from the square area. Knowing if the ISS is inside the box or not.

After that, once the ISS position is inside the area (which means that the distance between the desired point and the ISS is less than 630 km in Colombia and 450 in Europe) the code converts every time the distance from the ISS and the desired point into km and afterwards, checks if the distance is below 336 km (FOV radius of the ISS). Knowing, this way, when the ISS is focusing to the desired coordinates.

Finally, comment, that in the case that the distance is under 336 km, the developed program saves the time when the ISS has started focusing to the desired coordinates. Besides, explain that the program also realizes when the ISS exceeds from the 336 km, saving the exact time when the ISS has finished focusing to the desired coordinates too.

As a result, the use of the boundary area of the desired coordinates, allows an easy rejection for the further points and improves the speed of the program at the same time.

This way, a schema of the process is presented below :

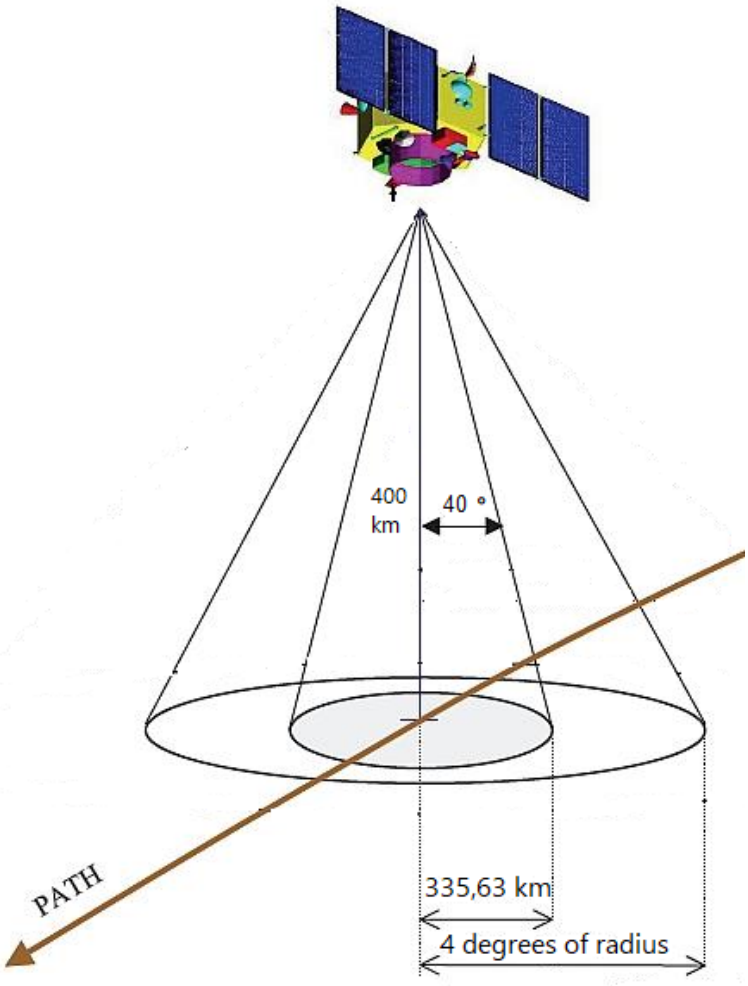

Figure 15: ISS FOV calculation (Adapted from [18])

After the brief explanation of how the system works, let's start with the work methodology of the main part of the code. The one that is focused in calculating and comparing the distance between the ISS and the desired point. Also the one that saves the starting and ending time of the FOV. This way, the followed steps are:

- 1. Calculus of the position of the ISS. As it has been explained previously in section 10.3.

- 2. Compare if the position of the ISS is inside the approximation area. In case it is not, reject it. Other way, calculate the distance between the ISS and the desired point in km for checking the 336 km limit.

- 3. Once the ISS is below the 336 km distance, the program checks if it is the first time in the whole simulation that the ISS crosses the circle area. If actually it is, the program saves the time in the results TXT file, the one that will be sent to the researchers, and it saves it as a variable too. It is also important to save the following times while the ISS is inside the 336 km circle area as an another Python variable. This way, as the only needed times are the ones when the satellite starts and ends focusing to the desired point, the following steps are done :

  - 3.1 The first value when the ISS crosses the 336 km area is saved as a variable and in the results file too.

  - 3.2 The times while the ISS is being inside the 336 km area, are saved in a Python variable, but are not recorded in the TXT. However, it is important to differentiate between the actual jump time and the last one (while the ISS is inside the 336 boundary contour) saving them as two different variables. Later the reader could understand why. Having this way 3 different variables: first cross, last jump and actual jump.

- 3.3 This way, the TXT time variable and the actual time variable are compared, in order to check if they correspond to the same orbit or not. A brief explanation is that one orbit corresponds to an hour and a half, so if the time between these variables is much less than 90 minutes, this will mean that both times correspond to the same orbit.

- 3.4 In the case the actual time jump inside the 336 km area and the first cross have a time difference bigger than 90 minutes, this will mean that they correspond to different orbits. Provoking that the last jump time done inside the 336 km area (the third variable) becomes the last FOV orbit time, which is the last time when the ISS was focusing to the desired point during the last orbit. Being necessary to save it in the final results TXT file as the ending time. Moreover, this also means that actual time corresponds to the new orbit FOV starting time, being necessary to save it too and starting, for the new orbit, the process again.

This process is applied as a loop in every single iteration time, when the program calculates that the ISS is under the 336 km distance.

However, there is a small possibility that the ISS makes only one step inside the circle area. For this reason, imagining this situation, the example of the values saved in the results TXT file for this three consecutive orbits will be the following:

Times:	Starting FOV time	Ending FOV time
First FOV orbit gap time (+1 steps)	05:30:00	05:35:00
Second FOV orbit gap time (1 step)	07:00:00	05:35:00
Third FOV orbit gap time (+1 step)	08:30:00	08:35:00

Table 9: Recorded step results demonstration (Version 1)

This happens attended that when a single step jump is recorded during an orbit, the variable corresponding to the last time when the ISS was focusing to the desired point (the going out time of the 336 km circle area) corresponds to the one of the last saved orbit, as this new orbit is only making one iteration jump that is being recorded as the beginning of the FOV period gap, not the ending.

In order to solve this problem, it was necessary to compare the last time value of the last orbit and the one supposed to be the exit of the actual orbit period. So, in the case they are equal,as it happens in the previous example, only the starting time must be recorded and not the exiting. Avoiding repetitions and correcting the problem.

This way, It will be achieved a correct result TXT file expressed as :

Times:	Starting FOV time	Ending FOV time
First FOV orbit gap time (+1 steps)	05:30:00	05:35:00
Second FOV orbit gap time (1 step)	07:00:00	- - - - - - -
Third FOV orbit gap time (+1 step)	08:30:00	08:35:00

Table 10: Recorded step results demonstration (Corrected version)

The following flow chart correspond to the part explained above, which is the one dedicated to check if the ISS is below the 336 km distance ratio to the desired point. However the code, placed in the annex, also includes the part of saving the first value of the whole simulation.

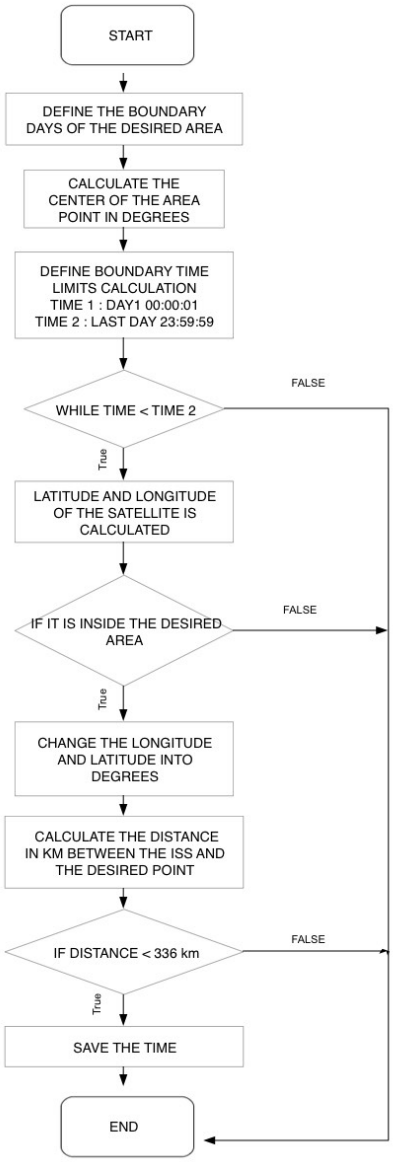

Figure 16: Flow chart 3

Calculation if the ISS is below the 336 km boundary flow chart

The equations used to convert into km the latitude and longitude distance between the ISS and the desired point, can be found in the annex.

The methodology of work will be explained in detail after this 4 points. But before that, the summary of the different TXT files used to write the time data in order to work with it. Hihhlight that every single file, except the one that includes the final results, will be crushed and over written in every single step jump. Just in order to reduce the memory needed. This way the following txt files are used:

- 1. Results : It will be the file to safe the first and last value of time when it crosses the FOV area. The results file, the file that will be send to the researchers.

- 2. Actual time ISS : It will be the file to read always the exact processing time of the ISS, to work with it. This file is created just to compare the different characters of the actual time inside the area (year-day-hour, remember that between orbits should pass 90 minutes more less. This means an hour or day change). This way is known if the actual step pertains to the same orbit or not.

- 3. Time : It will be the file used to save the first step jump of each orbit inside the area. In order to compare the time when the ISS was exiting of the area during the last orbit, and the actual time ISS exiting of the orbit. With the objective to not repeat the exiting time in the case the ISS has been only one step jump inside the area, as it has been explained in table 10 and 11.

- 4. Last time : It will be the file to save the time values while the ISS is crossing the area. In order to save the last one only.

So, after the first time the ISS crosses the area, highlight that the program is always comparing the last recorded time with the actual time, just to know if It changes of orbit or not. The steps of the main explanation are the following ones:

- 1. The actual time of the ISS is read and written in a file called "actual time" in order to be able to operate with it.

- 2. The first pass of the actual orbit inside the area, saved in the file "time" and the actual time saved in the file "actual time", are compared in order to know if they belong to the same orbit or not.

- 3. If they belong to the same orbit (they have the same year-day-hour composition), the actual time is saved It in the file "last time" where all the FOV step jumps are saved.

- 4. If they do not belong to the same orbit, It means the ISS has changed of orbit. So the last step time saved ("last time" file) that will correspond to the exiting time of the orbit happened before. And last results time written ("results" file) that will correspond to the starting time of the last orbit too, are compared. In order to check if the exiting time and the starting time of the last orbit are the same (1 step jump orbit), avoiding the problem of repeating exit times as It has been showed and explained in tables 10 and 11.

- 5. If they are exactly the same time, the exit time variable is not written in the results, as It will belong to the one happened two orbits orbit. If they are different, the exiting time of the actual orbit, and the starting time of the new orbit are written in the results file. Also the starting orbit one is written in the "time" file.

- 6. Finally the 5 seconds of the iteration are added. And it is passed to the next iteration loop.

Consequently, the flow charter of the actual part of the code is exposed in the next page. However, highlight that the actual lines for this part, can be find in the annex. This way, it is gotten:

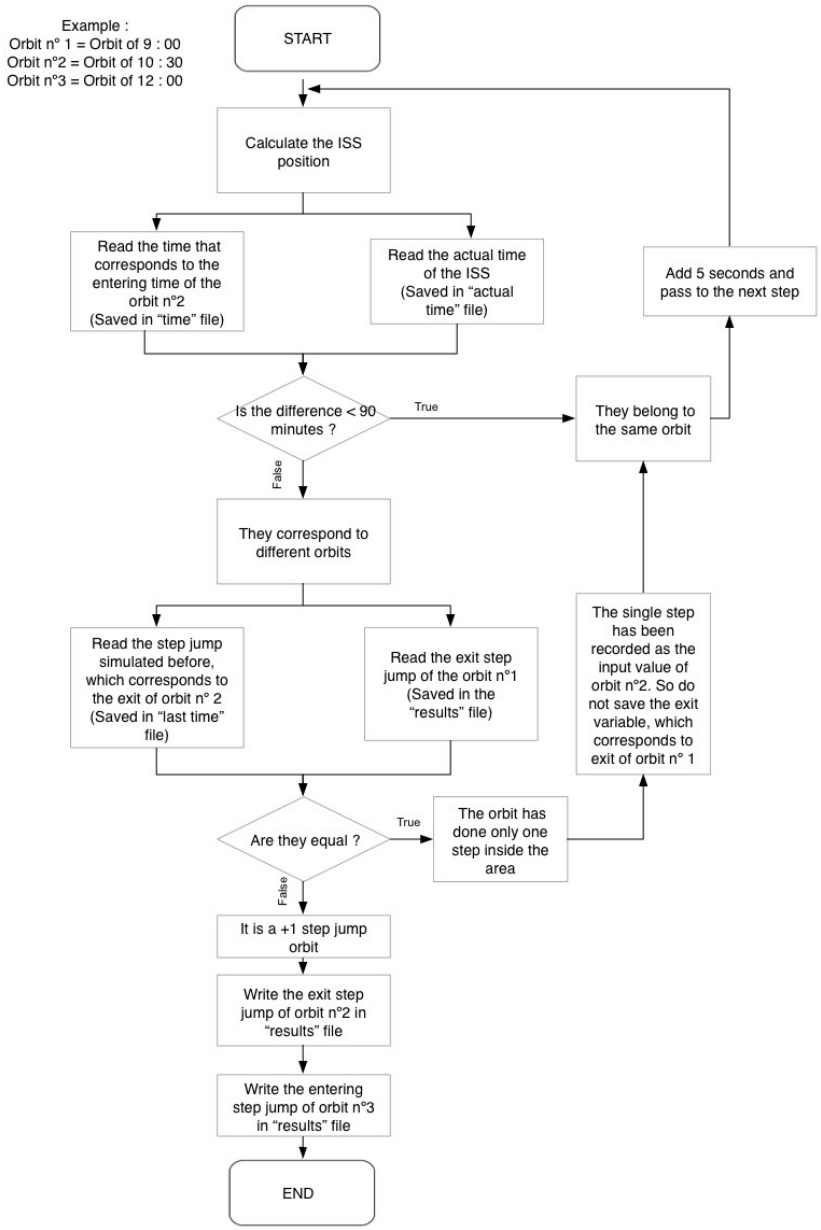

Figure 17: Flow chart 4

Main algorithm to compare and save the results flow chart

12.6 Send the results by e-mail

One interesting point that this code has, is that once the iterations have been realized and the whole simulation has been finished, the result TXT file is able to be sent by e-mail to the desired researchers.

Highlight that it is very useful to use this part of the code while the researchers are doing foreign campaign missions or if an automatic process is wanted too. This way, the following steps should been implemented:

- 1. Introduce the email to send the results file, the email that is going to receive the files, the filename of the results file and attachment body with a message, if it is desired. As a normal email. For this reason, a LRG e-mail was created.
- 2. Read the file.
- 3. Encode the file.
- 4. Connect to the server. For doing this, the password of the email should be introduced.
- 5. Send the email with the file.
- 6. Close the server.

This way, the code for sending the email was partially extracted from the website [19] and accordingly modified and adapted for the developed code. As a result, it was achieved the code placed in the annex.

12.7 Verification

As It was explained in section 10.3, there were three methodologies analyzed in order to check if the program worked well or not. Two of them, the first and second, were already explained in section 10.3. Which were to use the web ISS TRACKER plus the HDF file recieved from the UV researchers in order to check if the simulated/predicted Nadir position coincide.

However, on the other hand, the other methodology used to check if the program worked properly, was to use the LIS lighting information to know if the lightnings produced in a chosen area for many days were included in the FOV gaps that the program predicted. This way, as the LIS data shows with accuracy the data seen by the ISS, all the lightnings must be included inside the FOV gaps. For this reason, some random days were simulated for the location of Barranca, Colombia (latitude = 7,5 / longitude = -73,5) and afterwards, the lightning data for those days was carefully checked. Here, some examples are presented:

LIS lightning data date	Simulation FOV starting time	Simulation FOV ending time
2018-207 21:04:44.71	21:04:41	21:06:16
2018-235 21:13:26.07	21:13:16	21:14:36
2018-241 07:44:11.41	07:43:26	07:45:01
2018-269 07:51:25.56	07:51:16	07:52:46
2019-047 22:45:34.55	22:45:16	22:46:41
2019-075 11:46:06.49	11:46:01	11:47:31
2019-082 09:02:28.34	09:01:56	09:03:26

Table 11: Lightning validation data

It is important to comment that the test location was chosen because it has the LMA data. Being able to check the LIS data if it was necessary. This way, as a final result, all the lightnings were included inside gaps. for this reason, the program was considered to work properly. Before finishing, highlight that if the reader wants to visualize the whole code with comments, more information and other details, it can be done in the annex 1.

13 Code 2: ISS FOV calculation

13.1 Introduction

The objective of the code designed in this chapter is to be able to simulate, in a certain exact time, the exact ISS FOV cone and the Earth geometry. Extracting, this way, the exact coordinates from the intersection contour between both geometries and getting, as a result, the FOV upon the Earth of the different instruments that the ISS includes. This way, the main difference between this code and the program explained before, is that the last code is used to calculate the gaps of time where the ISS would be focusing to an exact coordinates, forgetting about the FOV contour. Using this new code for simulating the exact FOV contour in an exact second of a specific day previously chosen, in order to be able to georeference the lightnings detected by the MMIA sensor. However, this will be deeply explained later.

So before starting, it is important to specify some interesting facts about the information and the system axes that will be used.

13.2 HDF files

In order to simulate the ISS FOV in a certain time, two parameters should be known in advanced, which are the position and head direction of the ISS.

This information has been extracted from different HDF files, which are files that include all the ISS position and velocity information for a whole orbit and that can be easily found in this web [20]. Also, explain that the units for the position and velocity of the HDF files are the international system ones and that the HDF files, have their coordinate system centered in the middle of the Earth. Being the position and velocity expressed from there.

Comment, as an extra information, that the time given by the HDF files is the TAI time, so it should be transformed into normal time. TAI time is a high-precision atomic coordinate time standard based on the notional passage of proper time on Earth's geoid [21]. It is the principal realisation of Terrestrial Time (with a fixed offset of epoch).

This way, for extracting the ISS information from the HFD files for a certain time, a code previously developed by the researcher Joan Montanya has been used. Highlight that the correspondent part of the code, could be found in the annex.

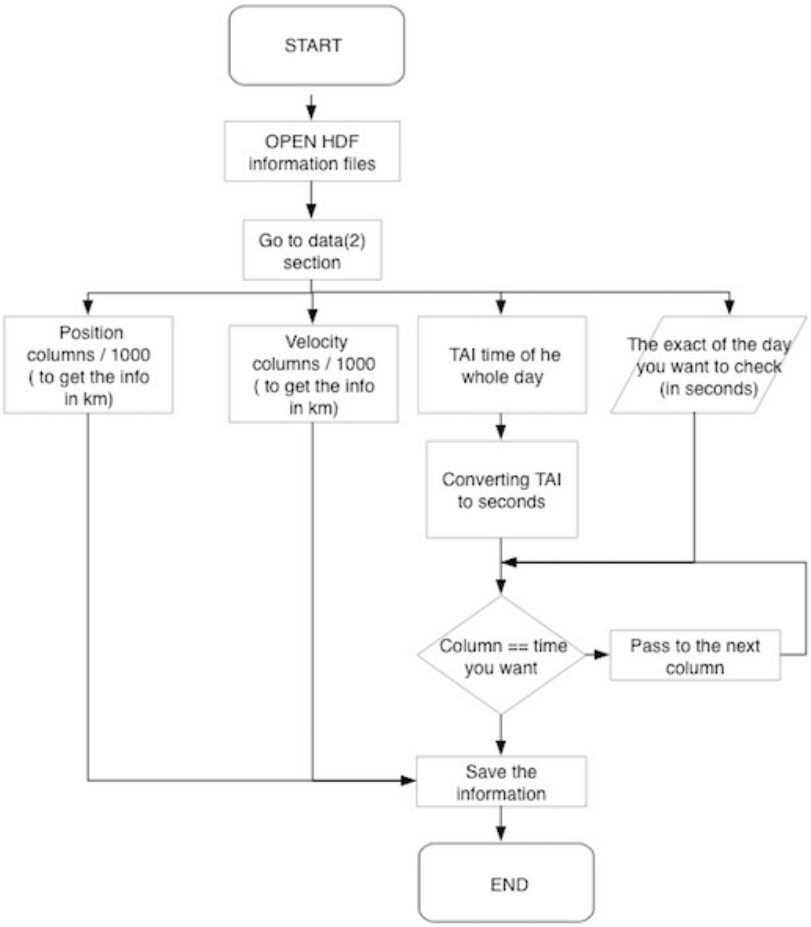

Figure 18: Flow chart 5

Main algorithm to extract HDF file information

13.3 Coordinate systems

As in this project is necessary to work with two different objects, the ISS and the Earth, this means that at least two different systems axes should be used. Obviously, as the final coordinates values from the contour must be expressed in degrees of latitude and longitude, this provokes that the final system of axes should be the one positioned in the center of the Earth. However, as the main objective of the code is to simulate the FOV of the instruments inside the ISS, it is prefered to start working with the coordinate system of the ISS. Placing ourselves above the ISS, as if we were taking a photo. For this reason, as it will be seen, many transformation matrixes should be applied in order to pass the contour coordinates from the ISS axes to the ones used for the Earth.

System of axes used for the Earth

The ISS has got many different coordinate systems [22], being used for different aspects. However, for the purpose of providing a common way to work, some historical standard reference axes have been designated during the years. One of these standard systems is the CTRS, used by the HDF files, which is the reference system used by the Global Positioning System.

The CTRS is an updated Earth-fixed system that incorporates polar precession. CTRS assumes a spherical Earth and does not take any flattening factors into account. The pole of this system is known as the CIO.

In this system, the Z-axis is coincident with the Earth's principal rotational axis, with the positive direction toward the CIO. The X-axis passes through the intersection of the CTRS reference equatorial plane and the CTRS reference meridian. The positive X-axis is in the direction of the CTRS reference meridian. The positive Y-axis completes the rotating right-handed Cartesian system.

A picture of the CTRS system is shown below as a schema:

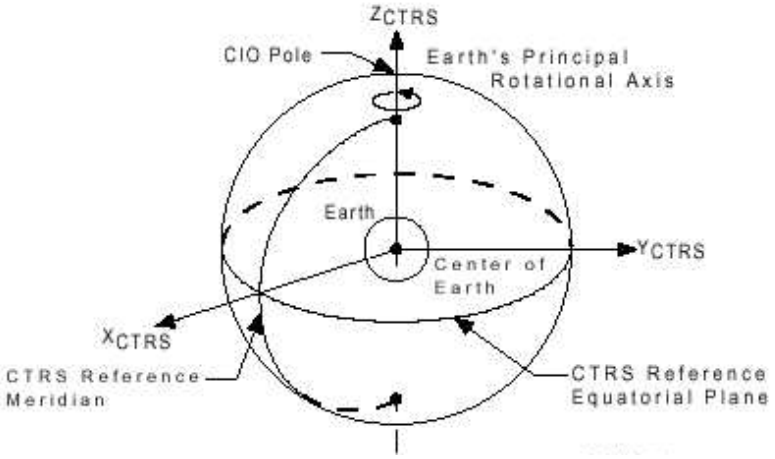

Figure 19: CTRS system [22]

For understanding the code, it is very important to know and understand exactly each change of coordinates done. For this reason, the following part is the most important one of the chapters, giving detailed explanations and showing many detailed pictures.

Starting and ending systems

As it has been said in the introduction, the process will simulate the FOV cone and Earth geometry, starting from the ISS axes and considering these ones fixed. The first reason why this is prefered, is because of the deviation that pitch, yaw and roll produce. If these parameters do not exist, the Earth will be perfectly positioned above the ISS. However, as these values provoke a relative movement of the Earth from the Nadir axis, the Earth is displaced to one side of the Nadir. This is the main reason why the ISS axes have been considered as the fixed ones, because moving an sphere in a coordinate system is much easier than moving a cone.

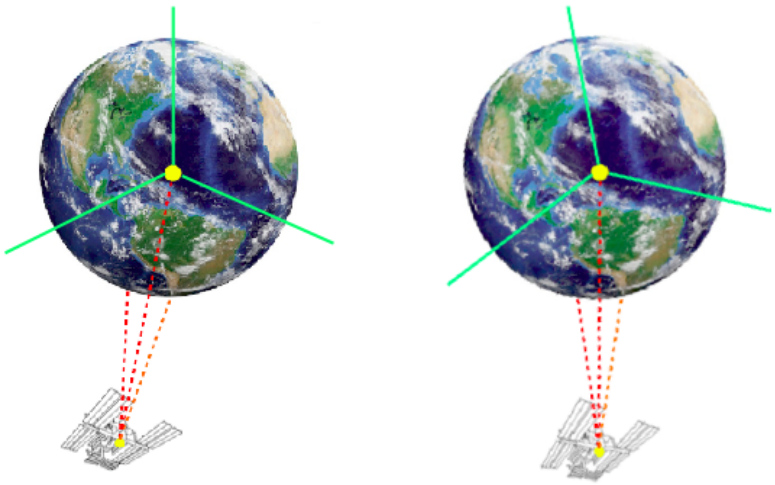

Figure 20: Example of rotating the FOV cone VS rotating the Earth

As it can be seen in the picture, working with the ISS axes for calculating the intersection between both geometries, avoids the need of moving the cone. Allowing the fact of using a simple grid, as the equations for a sphere are more simple than for a cone. However, the hard work will come to achieving the correct transformation matrixes.

Changes in the coordinate systems

First of all, remember the sytem of axes used for the ISS. Previously explained in section 10.

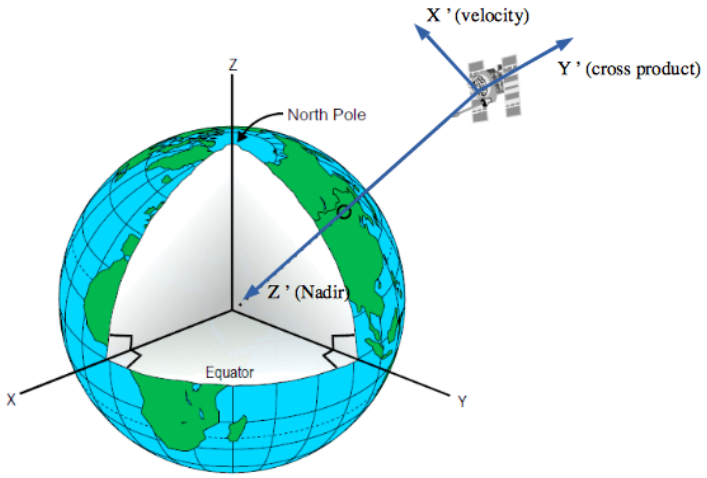

Figure 21: ISS and Earth systems (Adapted from [23])

Highlight that in order to achieve the transformation matrixes that allow to pass from the ISS system to the Earth one, the ISS axes should be expressed from the system of the Earth. However, as we are starting above the ISS, the sign of the Nadir vector extracted from the HDF files should be changed. Obtaining as a result a vector that starts in the ISS and finishes in the center of the Earth, the Z vector of the ISS system. Moreover, highlight, that the velocity vector must not change and that the Y axis is achieved making the cross product between Z and X axes.

Therefore, in order to not have problems with the dimensions, all the transformations will be done with unitary vectors. Although, this information was previously detailed in section 3.3.

Furthermore, the reader could not forget that the ISS has a deviation provoked by the pitch, yaw and roll parameters. However, it is quite difficult to pass from the pitch, yaw and roll system of axes to the Earth ones. So, in order to make transformations easier, many intermediate steps have been included. Achieving, this way, easier transformations.

This way, a summary of the different coordinate systems used for getting the latitude and longitude coordinates of the contour intersection, is presented below:

- 1. The simulation process starts with the pitch, yaw and roll coordinate system. Simulating the 40 degrees FOV cone in a vertical position and moving the sphere that represents the Earth over the space.

- 2. After that, the contour intersection between both surfaces is extracted, and its coordinates must be transformed into the ISS ideal system (the system without pitch, yaw and roll) using an Euler rotation matrix transformation.

- 3. Following that, those coordinates must pass into the coordinate system of the Earth, using a transformation matrix . However, in this case, this coordinate system has its origin in the ISS.

- 4. As a result of the before step, a translation should be applied to the coordinates in order to get them in a system centered on the Earth.

- 5. This way, finally, the latitude and longitude can be calculated. Using some equation that transform X,Y,Z vectors into latitude and longitude degrees.

A representation of all the coordinate systems, is presented:

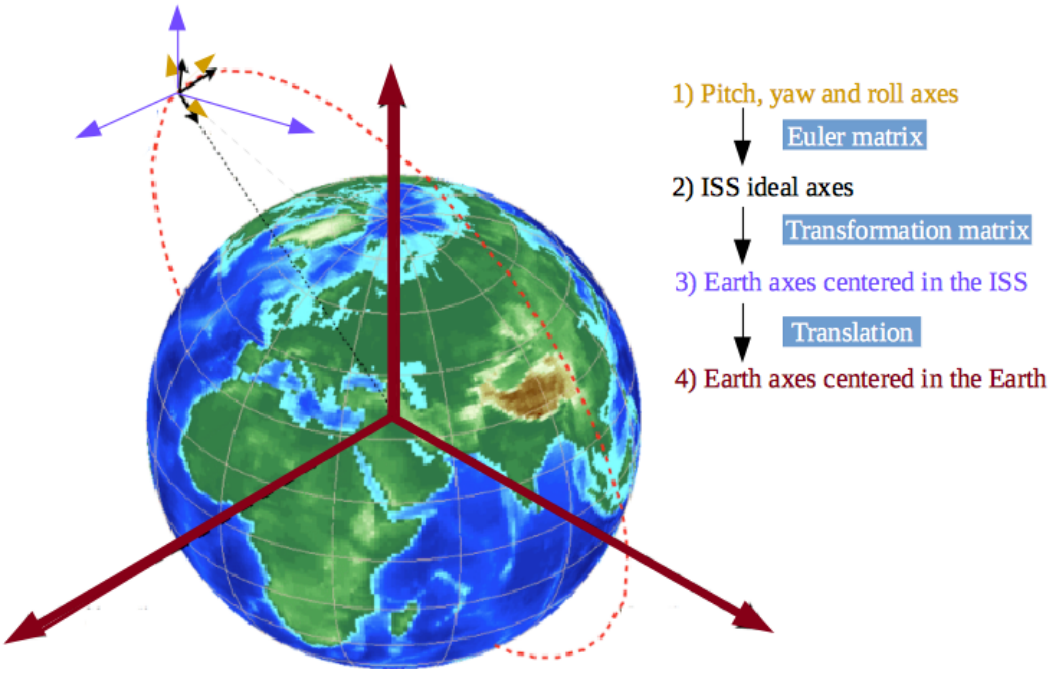

Figure 22: Summary of the coordinate system process (Adapted from [24])

So as it can be seen, two transformation matrixes and one translation are needed to convert the intersection contour coordinates between both geometries, into latitude and longitude degrees values.

This way, as it has been explained previously in section 3.3, for changing from pitch, yaw and roll system to the ideal ISS coordinate system, the inverse of the Euler rotation matrix (the one presented below) is needed. In Matlab, the inverse of a matrix can be easily achieved using the command "inv(Matrix)".

Understanding : Yaw $= \psi$, Pitch $= \theta$, Roll $= \phi$.

$$\begin{pmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi) & sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi) & sin(\phi)cos(\theta) \\ cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi) & cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi) & cos(\phi)cos(\theta) \end{pmatrix}$$

For the translation process, as both coordinate systems are parallel, the vector of the ISS position expressed from the Earth should be added to the FOV contour coordinates expressed from the ISS.

Here an example is presented:

Contour node position expressed from the ISS	(10,0,0)
ISS position expressed from the system of Earth	(- 100, 100, 100)
Final contour node position expressed from the Earth	(- 90, 100, 100)

Table 12: Translation process

Later, for passing from the ISS ideal coordinate system (the one without pitch, yaw and roll) into the coordinate system of the Earth placed in the ISS, it is necessary to calculate the transformation matrix between both vectorial systems.

The methodology of a vectorial space change process can be easily found in any algebra books. However, here a summary flow chart is presented. Highlight that the process described is to achieve the transformation matrix from the Earth system into the ISS system. So for getting the opposite process, the inverse of the final matrix must be done.

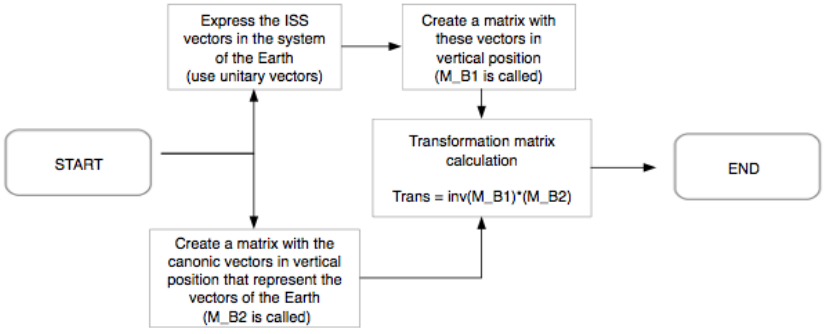

Figure 23: Transformation matrix calculation

13.4 Grid definition

In this section, the process followed to simulate both surfaces and extract the intersection contour will be explained. However, as the FOV cone and the sphere will be described using an explicit equation, a grid should be defined in order to get the X and Y coordinates for doing the surfaces simulation.

For calculating the intersection between both geometries, the Matlab "contour" function [25] was used. This function, requires to use one cartesian grid for all the geometries. However as the main object, speaking about size, is the sphere that represents the Earth, the best option would be to use a polar discretization system, multiplying later the coordinates for the cosines and sines of each angle, achieving this way a polar discretization in the cartesian system. Here a representation of the final discretization is presented :

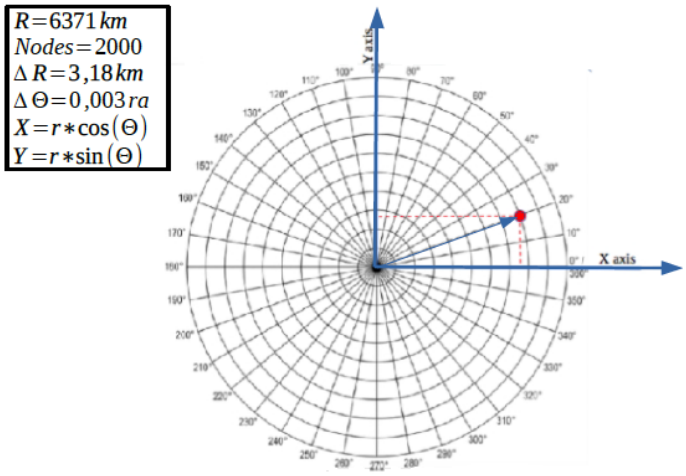

Figure 24: Discretization grid

It is important to comment that as the sphere is symmetric, it has two nodes for the same X and Y values (corresponding to the top and bottom parts). In order to use the contour function, a global matrix which includes the grid for the top and bottom parts of the Earth and the cone must be created. Being the last one included in the grid, because the size of the Earth is much bigger than the cone. Finally, comment that the code designed for the grid and the Earth's discretization is included in the annex.

13.5 Geometries simulation

1. Sphere

For calculating the intersection between both surfaces, first the grid of the sphere should be moved to the projection point of the center of the Earth. Right now, the origin of the grid for the simulation is placed in the (0,0,0) point, which is the position of the ISS. However, as it has been said before, because of the pitch, yaw and roll deviation degrees the Earth must be moved, as it is easier to simulate a vertical FOV cone.

For this reason, it is needed to multiply the Nadir vector (0,0,height) with the Euler matrix explained before. Adding those values into the cartesian grid coordinates. Making this way a translation of the grid into the projection of the center of the Earth and arriving, this way, to the final grid system. Which also includes the FOV cone inside.

A picture is added in order to make a clearer concept of the situation. So as it can be seen, the FOV cone will be for sure always inside the grid area. This way, the grid can be used for both surfaces.

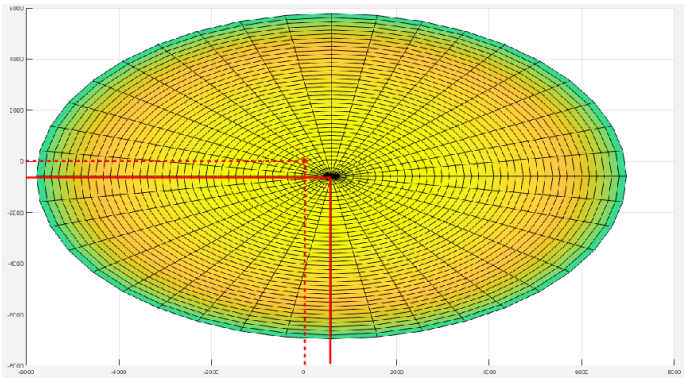

Figure 25: Displacement of the Earth from the top for a pitch = 5°, roll = 5° and yaw = 4°

As the CTRS system of the HDF files, consider the Earth as an sphere, the equation used for calculating the top and bottom parts of the Earth are:

$$Z = \sqrt{R^2 - (X_{grid})^2 - (Y_{grid})^2} \tag{1}$$

$$Z_{top} = Z + Nadir_{projection} \tag{2}$$

$$Z_{bottom} = Z - Nadir_{projection} \tag{3}$$

Here, a summary of the process explained in the last two sections is presented:

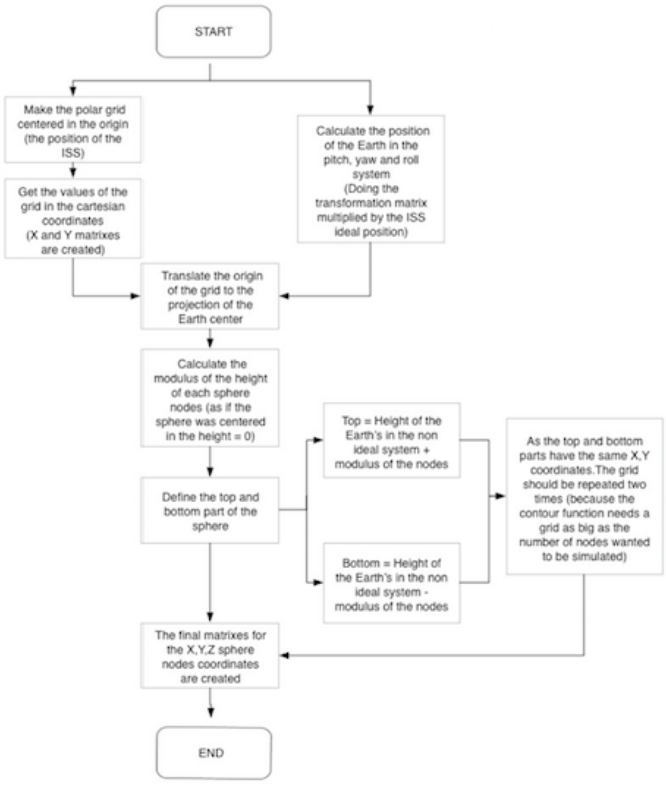

Figure 26: Earth simulation schema

Calculate the nodes of the Earth

2. Cone

As it has been said, the ISS FOV cone is included inside the grid area. This way the cone can be calculated with the same grid used for the sphere, as the contour function requires. This way, to calculate the Z coordinates for the cone, the equations used have been :

Sensor angle = s (In the case of the LIS = 40°)

$$a = (atan(\frac{s * pi}{180}))^2 \tag{4}$$

$$Zcone = \frac{\sqrt{(Xgrid)^2 + (Ygrid)^2}}{a} \tag{5}$$

Following these equations, the coordinates for the cone are achieved. Even though, as the grid is much bigger than the cone, the cone will over pass the sphere, obtaining this way, two connection surfaces. For this reason, as the only interesting intersection part is the one produced at the bottom, only the intersection coordinates produced in the bottom part will be saved.

Here it is an example of the final simulation :

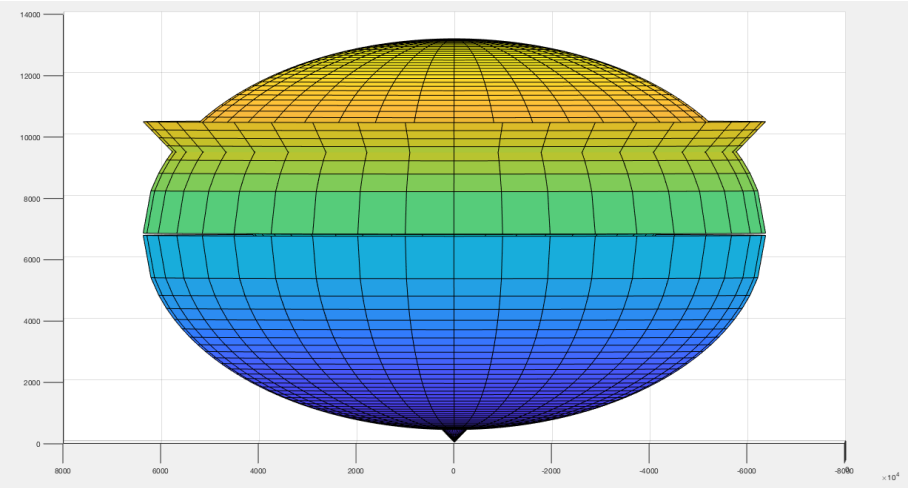

Figure 27: Simulation of the geometries

Afterwards, once the cone and the sphere nodes have been calculated, the Matlab "contour" function must be used to obtain the intersection X and Y coordinates. Highlight, that the contour function works similarly as the Newton method explained in section 10.3, because it calculates the intersection between both surfaces from the nearest nodes. So the methodology for obtaining the intersection nodes is the following :

- 1. Make the difference from both surfaces in order to achieve the nearest nodes.
$$Z_{diff} = Z_{Earth} - Z_{cone}$$
- 2. Apply the contour function for calculating the X,Y cartesian values when the connection is produced. This way, this command should be written.
$$C = contour(X_{grid}, Y_{grid}, Z_{diff}, [00])$$
- 3. Extract and save all the X and Y coordinates, from the nodes where the connection is produced.
$$XL = C(1, 1 : end) and YL = C(2, 1 : end)$$

The Matlab help website [26] has been used as a reference for the calculation of the intersection points using the contour function. Here, an example of the contour from the intersection parts is presented.

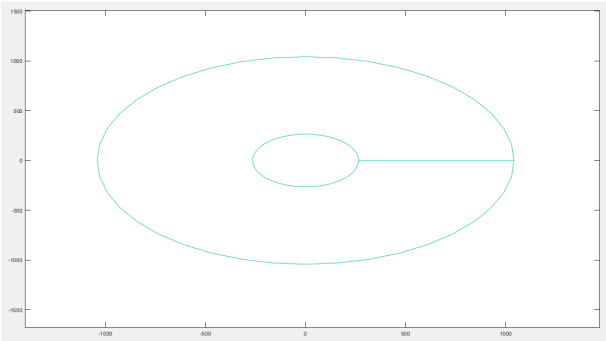

Figure 28: FOV contour intersection, top and bottom parts, seen from the top of the Z axis for a non pitch, yaw and roll case

13.6 Getting the final coordinates in latitude and longitude degrees

Once the intersection points between the Earth and the cone have been calculated, the process to express them in latitude and longitude should start.

For achieving this, as it has been explained previously, two space transformation matrixes and one translation should be applied. Just as a memorial, the steps needed to be followed are:

- 1. Pass from the pitch, yaw and roll system to the ISS ideal one, using the inverse of the Euler transformation matrix exposed in section 11.3.
- 2. Pass from the ideal ISS system to the cartesian one placed in the ISS, using the steps explained in figure 22.
- 3. Make a translation in order to place the connection points in the system centered in the Earth.
- 4. Finally, once the intersection points are expressed in the Earth cartesian system, the latitude and longitude conversion must be done.

The parts of the code correspondent to the transformation and translation steps are included in the annex with comments. However, it is only needed to multiply the X and Y definitive coordinates of the intersection, by the inverse of the Euler matrix and consecutively by the inverse of the transformation matrix. Applying after that the translation.

Finally, for expressing the results in degrees of latitude and longitude, the following equations should be applied:

- **1. Latitude:**

$$Latitude = (asin(\frac{Z}{6371}))\frac{180}{pi} \tag{6}$$

- **2. Longitude:**

2.1 If X <0:

$$Longitude = (atan(\frac{Y}{X}))\frac{180}{pi} \tag{7}$$

2.2 If X <0 and Y >0:

$$Longitude = (atan(\frac{Y}{X}))\frac{180}{pi} + 180 \tag{8}$$

2.3 If X <0 and Y <0:

$$Longitude = (atan(\frac{Y}{X}))\frac{180}{pi} - 180 \tag{9}$$

Highlight that those equations have been extracted from the programming website [27].

13.7 Verification

As this code will be used to complement and correct the researcher's work, the verification of the program must be done with the UV researcher's program in order to achieve if the solutions are equal or not.For this reason, as the UV researcher's code is only able to calculate the Nadir position of the ISS, the Matlab program developed in this TFG has been applied without any angle of pitch, yaw or roll. Achieving the Nadir position, simulating a cone of 0,1 degrees width.

Moreover, for calculating the exact latitude and longitude of the Nadir, the program should catch only the position of the intersection nodes placed at the bottom part of the Earth and discard all the others from the top part. Hence, as the distance of the ISS from the center of the Earth is known (the modulus of the Nadir vector) and the Earth radius is also known (6371 km), the program will only save the intersection coordinates values placed in :

- Z coordinates = Nadir modulus - (6371 +/- 0.1)

Here a schema is presented:

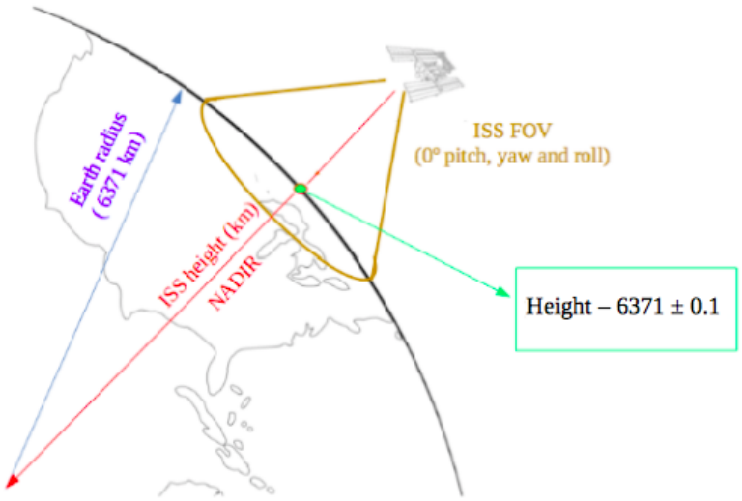

Figure 29: Save the projection of the Nadir (Own figure)

Following the above explanation, several points have been tested and hence, several statistics have been achieved. So, it has been achieved:

Date	Second	Valencia Lat.	Program Lat.	Valencia Lon.	Program Lon.	Error Lat.	Error Lon.
2018/06/27	69946	-2,0927	-1,8553	116,8295	116,9848	0,2374	0,1553
2018/06/29	30704-30705	56,4576	8,6459	33,3526	-80,2806	/	/
2018/06/30	72247-72248	16,0265	16,1036	103,9577	104,1457	0,0771	0,188
2018/07/03	24039	8,8823	9,0512	-75,781	-75,592	0,1689	0,189
2018/07/13	60433	23,5813	23,4714	83,4484	83,5637	-0,1099	0,1153
2018/08/13	6912	6,7501	6,5255	-80,4413	-80,3254	-0,2246	0,1159
2018/08/30	2516	4,2857	4,4222	25,4775	25,6057	0,1365	0,1282
2018/09/01	29993	16,5338	16,5374	-92,673	-92,539	0,0036	0,134
2018/09/05	6671	16,7581	16,7589	-17,5622	-17,4281	0,0008	0,1341
2018/09/25	31270	11,8796	11,8055	-84,4883	-84,52	-0,0741	-0,0317
2018/10/05	51196	-6,1392	-6,0662	145,8865	145,8768	0,073	-0,0097
2018/10/06	19900	17,3732	17,206	-106,7177	-106,7205	-0,1672	-0,0028
2018/10/10	46892-46893	3,3189	3,3328	126,7694	126,7363	0,0139	-0,0331
2018/10/11	49404	3,225	3,2246	110,236	110,2138	0,0004	-0,0222
2018/10/25	76412	8,4279	8,3106	101,5539	101,5278	0,1173	0,0261
2018/10/25	76452	10,4225	10,3107	103,0212	103,0135	0,1118	0,0077
2818/11/02	18571	3,573	3,543	-65,0902	-65,0868	0,03	-0,0034
2018/11/28	69713	-4,6831	-4,6379	103,9628	103,9629	-0,0452	0,0001

Figure 30: Results of the tests

Firstly, as it highlighted in the table, it can be seen that the Valencia code is not 100% efficient, getting one wrong value comparing with my code and the ISS tracker. It is important to say that the Valencia code extracts its information from the ISS data, so if one day the ISS is processing wrongly the data, as a consequence the code will work wrong. An example of this is the test of **2018/06/29** during the seconds **30704-30705**, where the Nadir position achieved using the ISS tracker is **(Lat = 8,7 and Lon = -80,28)**. Getting, using the ISS tracker value, a difference of **Lat = -0,0541 and Lon = -0,0006** degrees with my program.

As a summary, it can be seen that for the latitude, around the 90% of the tests give an error under or around 0,1 ° (56% below and 34 % around 0,1). While in longitude the 100% of time give an error under or around 0,1 ° (56% below and 44% around 0,1). Considering, this way, that the code is validated.

Finally, as the program developed to calculate the FOV of the ISS is done, the next step will be to try to georeference the lightnings with it.

14 Georeference of lightning flashes

In this section a methodology for the georeference of lightning flashes has been developed. However, in order to be able to do it, the user should use the MATLAB and QGIS softwares. The last one with the georeference plugin, which can be easily found in [28].

This section will explain in detail the designed process, hypothesis and calculations that should be followed for georeference the LIS lightnings seen by MMIA. Moreover, an analysis of the divergence between the results and the real lightnings will be done. Exposing, finally, an example. However, comment that for testing the program 11 cases have been analyzed in detail, including the final results in the annex.

This way, the methodology followed is :

14.1 Retrieving MMIA camera frames

The first step is to run the Matlab code developed by the UV researcher. This program, allows to get the pitch, yaw and roll of the ISS for an exact time. However, it is important to highlight that for using this program a CDF file, a file that include all the technical information recorded directly from the ISS, is required. Unfortunately, this CDF information is only available for the ASIM researchers and for this reason the code developed by the UV researcher is not showed in this work.

Nevertheless, the code developed by the UV researcher is not only used for achieving the ISS attitude. In addition, the code also provides some photometers analysis, besides the necessary MXGS and MMIA pictures (full size and with a zoom to the detailed lightning section) that are commonly used for detailed studies. However, the only picture needed in our case will be the MMIA full size one, in order to georeference the FOV of the MMIA sensor with the lightnings included.

In these two pages, an example of the output pictures is shown :

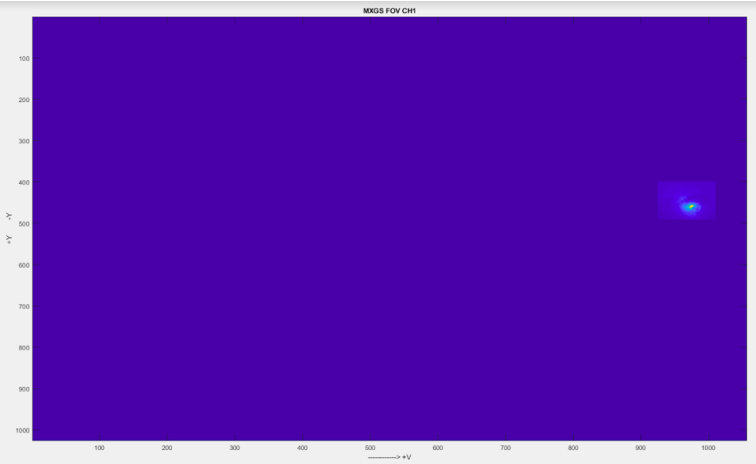

Figure 31: Output 1: MXGS detection

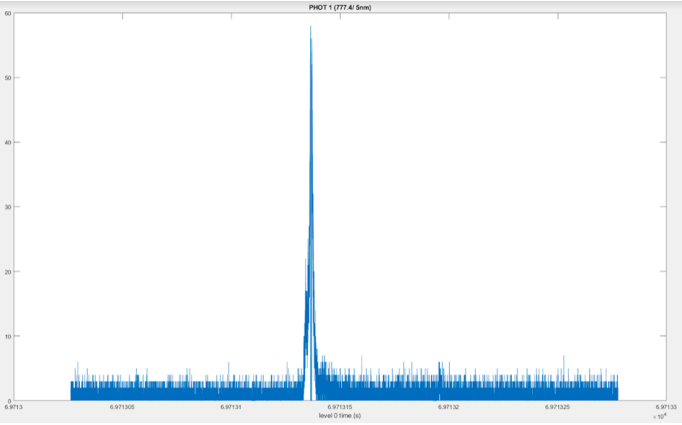

Figure 32: Output 2: Photometer analysis

However, these two pictures are the ones that will not be used.

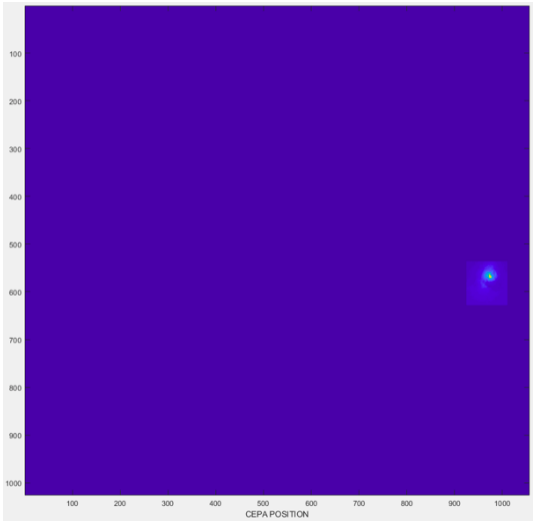

Figure 33: Output 3: MMIA detection

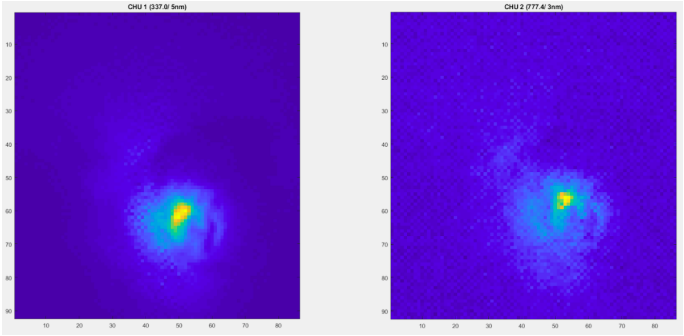

Figure 34: Output 4: MMIA detection for oxygen and UV channels (zoom version)

It is important to highlight that the MXGS and MMIA are not placed in the same position, this is the reason why the pictures are different. However, in case of the MMIA, the horizontal axis is in fact the X axis of the pitch, yaw and roll system. Finally, the output 3 will be the one used for georeference the lightnings.

14.2 Extract the lightning information

The second step is to extract the detailed information of the lightnings produced in a desired region and time. This step is done in order to be able to only represent the lightnings which can have been seen from the ISS, rejecting all the foreign ones.

Hence, the procedure is :

- 1. Search the desired HDF files (the files which contain all the information from the lightnings produced and seen by the ISS in the whole orbit) from [29], choosing the wanted region and period. Afterward, once the HDF file has been chosen, it should be downloaded from [20].

- 2. After that, the HDF information should be read and saved in a TXT file. In this case, a code developed for a researcher of the LRG has been used, achieving this item easily.

- 3. Once the whole orbit information has been saved in a TXT file, the lightnings placed in an interval of time about +/-1 second of the considered time, should be searched. Highlight that this step is produced because of the miss-match between the LIS and MMIA time, although a researcher of the LRG hass developed a technique (during the process of this TFG) that allows to correct the time difference between both sensors. Being able to select the exact lightnings showed in the MMIA photometers analysis, being them the ones showed in the pictures.

  Unfortunately, the time employed in the development of this TFG is limited and this program was not included in the actual process. Instead of this, all the flashes (+/- 1 second) were represented and an analysis of the intensity was done. Being the flashes with higher intensity, the ones which are shown in the MMIA picture.

- 4. The chosen flashes have to be represented. For doing this, a Matlab code that uses the function Worldmap [30] was developed. This function is used for constructing map axes for a given region of the world, achieving as a result, the representation of the desired flashes in a grid.

  However, an important improvement was implemented. Being able to represent the different groups of flashed with different symbology and plotting with different intensity, each lightnings in function of their radiance. Getting a stronger color intensity as the radiance increases, as it can be seen in the following picture :

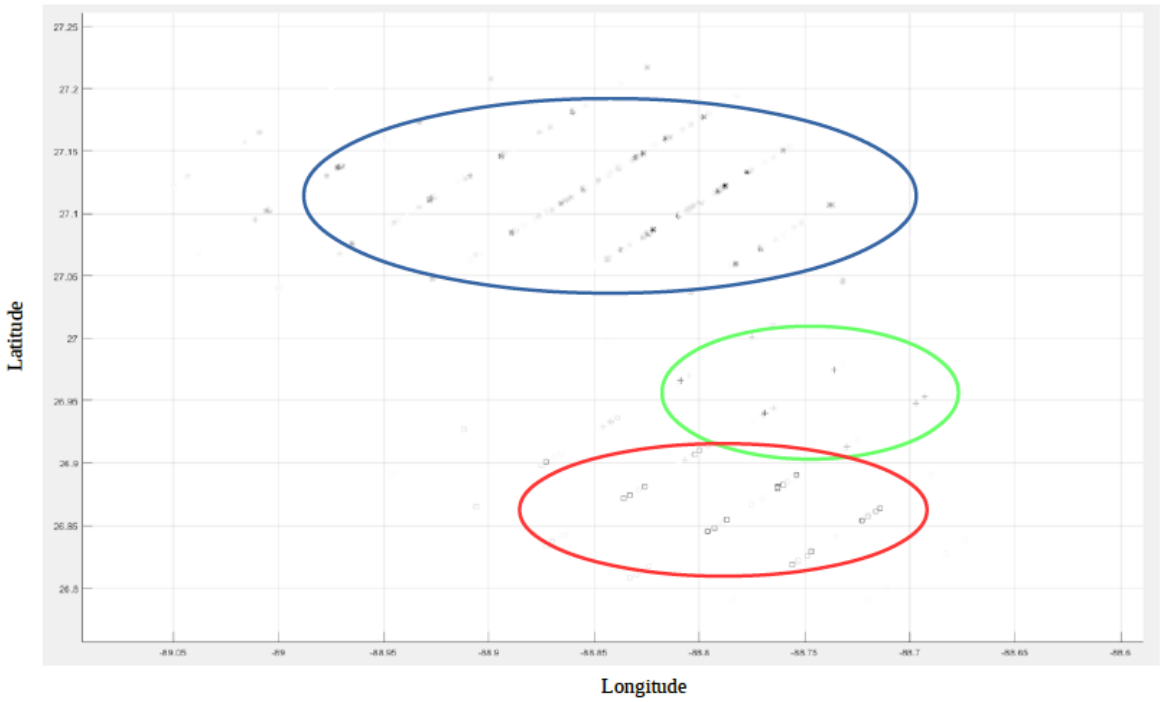

Figure 35: The representation of the flashes by its group and radiance for case 2019/05/5 06:52:25

Also highlight that the code used for this can be seen in the annex.

- 5a. Once the MMIA picture is extracted and the coordinates of the desired flashes are known, the next step is to calculate/simulate the FOV of the MMIA picture.

  In order to do that a new version of the FOV code, previously explained in chapter 12, was designed. However, as this part is the main one for georeference the lightnings, the considerations chosen will be explained in detail.

  As it could have been seen, the MMIA camera produces a square FOV picture, getting as a result a smaller FOV than the one achieved in the code of chapter 12. Being the FOV of the camera included inside the ellipse predicted.

  However, the fact that the camera produces a square FOV simplifies the calculation, because now it is not necessary to calculate the exact intersection between the Earth and the FOV of the ISS sensors. Being able to consider "the floor" in an horizontal plane placed at the height of the cloud, as it can be seen in the following picture, defining the process of a photo taken by a camera.

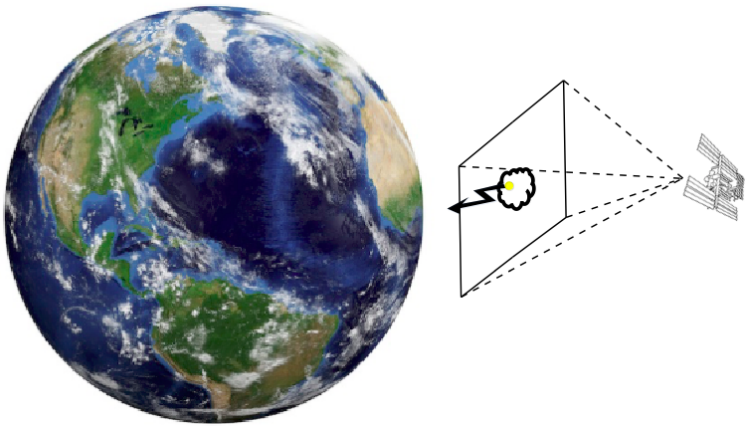

Figure 36: MMIA camera FOV example (Own figure)

It is important to define the parallax of a satellite, which is a common problem found in these cases. The parallax of a satellite, is defined as a displacement or difference in the apparent position of an object viewed along two different lines of sight. Below, an example of the parallax definition is presented :

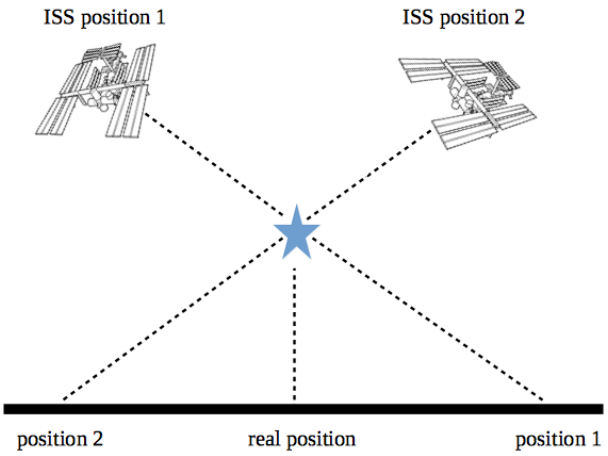

Figure 37: Parallax definition (Own figure)

As it could be seen, using the methodology of calculating the FOV in the cloud height against instead of the surface of the Earth, provokes that the main part of the parallax of the satellite is almost avoided. However, two problems are still present :

1. The first one is caused because of the lack of knowledge about the cloud height. Although, it has been considered the value of 10 km in Europe and 15 km in Colombia. These values were chosen because the researchers have demanded this way.

2. The second problem is caused by the pitch, yaw and roll of the satellite. Although without them the parallax problem would be corrected, the fact of having these degrees of deviation from the Nadir causes a little parallax effect. Provoking as a result, few km of mistake.

However, it is important to highlight that as the orbit of the ISS is very low, the parallax of a lightning seen from the ISS and projected to the Earth's surface will be about 10 km (a low distance if it is considered that the FOV of the ASIM instruments are about 680 km of width), reducing also this distance by considering the floor in the height of the cloud.

- 5b. It can be said that in order to georeference the lightnings, both codes explained in sections 10 and 11 are mixed. For this reason, it will be easier to understand the process if we place ourselves above the ISS, measuring the coordinates of the corners in the ISS pitch, yaw and roll axes system. Thereupon, the transformation matrix defined in chapter 11 for passing the coordinates into the axes system of the Earth, must be applied. Getting as a result, the latitude and longitude of the corners of the picture.

  Here it is shown an example of how the coordinates of the corners are defined. However, it is important to highlight that these coordinates should be extracted using the pitch, yaw and roll axes system.

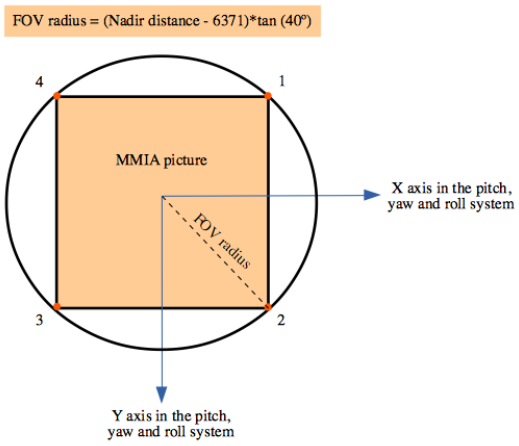

Figure 38: MMIA sensor FOV example

Only highlight that instead of the code developed in chapter 11, this code takes into account the pitch, yaw and roll of the ISS. Although it also considers the exact height of the ISS, as the Nadir distance is extracted directly from the HDF files.Getting as a result more accurate results.

This way, the coordinates of the corners are :

Corner	X	Y	Z
1	$\frac{+Radio}{\sqrt{2}}$	$\frac{-Radio}{\sqrt{2}}$	Nadir distance - 6371 - Cloud height
2	$\frac{+Radio}{\sqrt{2}}$	$\frac{+Radio}{\sqrt{2}}$	Nadir distance - 6371 - Cloud height
3	$\frac{-Radio}{\sqrt{2}}$	$\frac{+Radio}{\sqrt{2}}$	Nadir distance - 6371 - Cloud height
1	$\frac{-Radio}{\sqrt{2}}$	$\frac{-Radio}{\sqrt{2}}$	Nadir distance - 6371 - Cloud height

Table 13: Corners coordinates

Once the coordinates of the corners are achieved, the steps that focus to pass the coordinates on the system centered in the Earth applying all the transformation matrixes previously explained in chapter 12, should be carried out. This way, the latitude and longitude coordinates of the corners of the picture are achieved.

As a comment, highlight that the code used in this section can be achieved and seen in the annex.

- 6. Once that all the necessary information and pictures have been achieved, it is time to georeference the LIS flashes and the MMIA pictures. This step was done with the QGIS software, getting as result a deviation between the obtained lightning coordinates and the real ones, about 40 or 50 km. However, it is important to highlight that the UV researchers consider that the MMIA sensor has a deviation from its own Z axis about 5 degrees, which would provoke a deviation of 35 km between the obtained position. Reducing, this way, the difference to 15 km.

  Here an example of one simulation is shown, however, the result of all the examples can be found in the annex.

Figure 39: Georeference done for the case 2019/05/05 06:51:47

15 Conclusions

This bachelor thesis has three main parts: 1. Forecasting ISS passes / 2. ISS FOV calculation / 3. Georeference lightning flashes. This way, the conclusions will be made for each part, adding at the end a final summary of the project requirements, in order to see if they have been achieved or not.

1. Forecasting ISS passes:

This task consisted in predicting when the ISS has been and would be passing over a desired area. As a starting point, a code made in Python from an LRG researcher was used, being necessary to learn that language. However, as the code developed by the researcher did not work properly, it was necessary to restart and develop many improvements in order to get accurate results.

The most important change applied was applied to avoid the use of the Newton function, which provokes errors in certain orbits. For this reason, the code developed is able to simulate without any error the 16 orbits that the ISS produce, achieving the same results as the code of the UV researchers and the ISS tracker. Moreover, it was also obtained a fast computational time, as the code developed during this TFG is able to simulate 100 days in half a minute.

Following, the second most important change, was passing from a Gregorian calendar into Julian dates. This change, provoked that an historical TLE list could be used. Being able, as a consequence, to make historical observations in order to search the exact lightning data from LIS.

As a summary, the code has been checked using three different methodologies, obtaining in all three high accurate results.

2. ISS FOV calculation:

This task consisted to develop a code that is able to predict the exact FOV of all the instruments above the ISS. In order to achieve that, a simulation of the Earth geometry and the FOV cone of the instruments above the ISS, was done.

It is important to say, first, that this code is able to calculate the exact intersection between both geometries. Getting as a consequence, the projection of the FOV over the Earth. However, as the orientation of the FOV cone is defined by the pitch, yaw and roll parameters, the exact values are needed. In the case of this TFG, these parameters were achieved using the code developed by the UV researchers, however, they should be compared (if it is possible) with the real values from the ISS.

This way, in order to validate the code, the Nadir position has been used. Comparing the results of the developed code with the results from ISS tracker and the code of the UV researchers. Achieving an accuracy of less than 0.1 degrees of difference between the real value and the simulated one for over almost all the cases. Having an exact accuracy fro over the 90 per cent of the cases. Considering, this code valid for georeference lightning flashes.

3. Georeference lightning flashes:

Finally, a methodology to georeference lightning flashes using the above code and the MMIA pictures, was defined. For this part it is important to say that right now, the UV researchers think that the MMIA sensors can be deviated 5 degrees from the theoretical position (the one used during the thesis), which can provoke 35 km of error between the predicted position and the real one. Moreover, as it has been previously said, the pitch, yaw and roll values achieved from the code developed by the UV researchers, could be different from the exact ISS values. Being necessary in further studies, if it is possible, to compare this data with the real values.

Hence, as the unknown inputs deviation could produce at least 40 km deviation, having between 40 and 50 km of deviation in the actual process, the process defined can be considered a success.

As a summary, it is considered a good idea to check and revise if all the initial requirements and final objectives of this thesis were achieved :

- 1. Develop all the codes and processes in softwares that could be used by the LRG researchers and that can be easily sent to other researchers.

- 2. Develop a code that allows to predict when the ISS will be focusing to a desired coordinates.

- 3. Develop a code that allows to extract the FOV of the instruments above the ISS and develop a process that allows to georeference lightnings.

So, as it could be seen during this thesis development, the precision of the two developed codes is high enough to consider these requirements achieved. However, it is true that more accurate hypothesis would help to achieve better results in the process of georeference the lightnings, although for achieving that, the exact attitude of the ISS and the exact positions of the MMIA sensors are needed.

16 Next steps

In this chapter, some next steps are proposed in order to make a deeper analysis of the developed process. However, as this TFG is directly related with the work currently developed by the researchers, some improvements of the programs used were thought during the final weeks of this thesis, although because of the limited time I had, these ideas were not finally implemented. This way, the proposed improvements are :

- 1. Use the code developed by LRG researchers that allows to know, from the photometers, the exact flashes that should georeference.

- 2. Check if the code developed by the UV researchers, gives the exact pitch, yaw and roll values. Compare these values with exact ISS information from NASA, in order to correct them and use the real ones.

- 3. Improve the process developed in this TFG for georeference the lightnings, in order to represent in the QGIS software all the MMIA frames pictures. Having, this way, all the spotlighted zones.

- 4. Search and test and ideal case where the lightning it is inside the Nadir line. Having, this way, no pitch and roll and being able, as a consequence, to extract the exact error of the process.

- 5. Search and check if the MMIA cameras are deviated 5 degrees from the supposed position. This has been mentioned a few times during the thesis and it is a supposition that the UV researchers have. It will be a great idea to check if the MMIA sensor is deviated from the supposed position or not, provoking this way a deviation of 35 km in the georeference process (as the process simulates a prediction with the theoretical conditions).

- 6. Develop the exact ISS FOV code (chapter 12) with a pyramid that simulates the MMIA sensors. Right now, the code is simulating a FOV. Being able to simulate a pyramid, will avoid chapter 13.

- 7. Change the colors used to represent the lightning in the MMIA picture. The use of a new scale of colors in pictures extracted from the code of the UV researchers, could help to highlight more the lightning zone.

  As it can be seen in the next picture (which is a zoom of the real one), the dimensions of the LIS lightning detection and the one from the picture are similar. However, the highlighted zone is smaller, producing confusion. A change of the scale of colors will avoid that problem, as it can be seen:

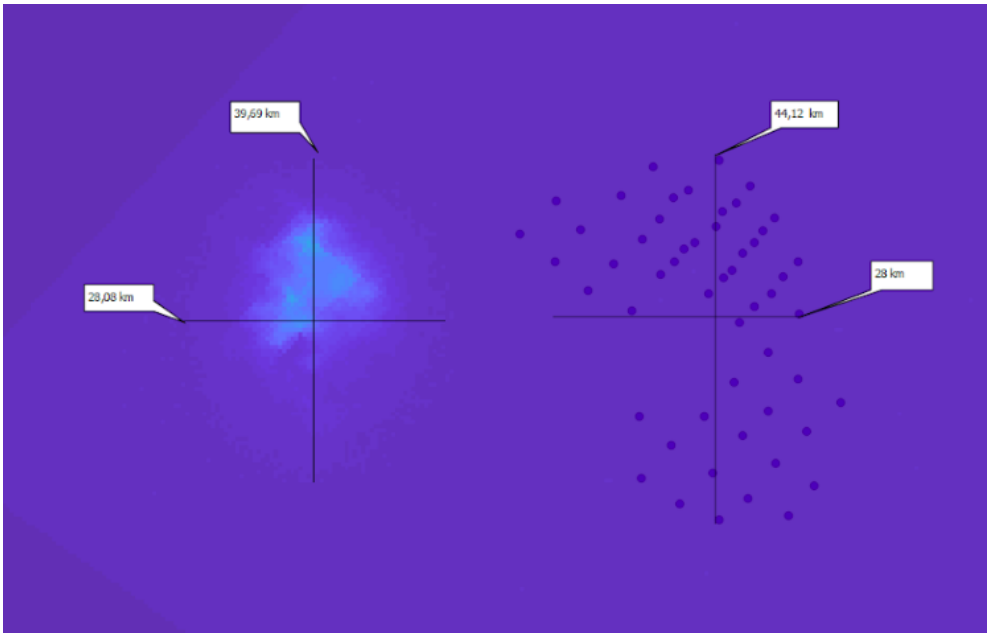

Figure 40: Actual scale of colors

In the last picture, it is represented the zoom of the georeference of the lightning flashes done for the case 2019/05/05 06:52:25. This case was used to check if the dimensions of the pixels, for both LIS and MMIA pictures, corresponded with their theoretical values. Achieving a positive answer. Highlight that the blue points are the representation of the flashes detected by the LIS in their exact latitude and longitude position, however, the left plot is the georeference of the MMIA picture.

Moreover, in this case can be seen that the representation of the picture from MMIA has the same area (speaking about sizing) as the LIS one. Which means that the georeference of the lightning flashes made from the MMIA pictures, does not modify the area of the flash.

Finally, comment, that if the colors for representing the MMIA pictures were changed, a better representation of the zone of the lightning flashes would be achieved. Identifying in a better way the lightning flashes, as it can be seen in the following picture :

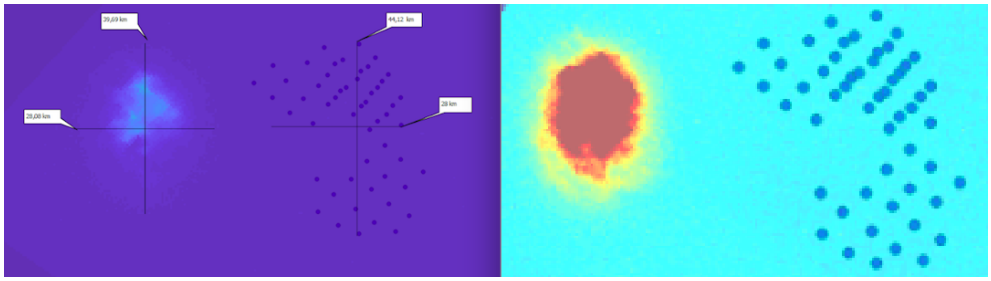

Figure 41: Actual scale of colors vs the one proposed

17 Bibliography

References

[1] ASIM. (2019). Retrieved from *https://www.uib.no/en/rg/space/53138/asim*

[2] International Space Station (ISS) Lightning Imaging Sensor (LIS) Provisional datasets. (2019). Retrieved from *https://ghrc.nsstc.nasa.gov/pub/lis/iss/doc/isslis$_d$ataset.pdf*

[3] ISS Utilization: LIS - Satellite Missions - eoPortal Directory. (2019). Retrieved from *https://directory.eoportal.org/web/eoportal/satellite-missions/i/iss-lis*

[4] López Trujillo, Jesús Alberto; Montañá Puig, Juan; Van der Velde, Oscar Arnoud; Romero Durán, David; Aranguren Fino, Harby Daniel; Torres Sanchez, Horacio; Taborda, John; Martínez, Joaquin (Zakon Group, 2016)

[5] Catalunya, U. (2019). LMA Network — Lightning Research Group. LRG — UPC. Universitat Politècnica de Catalunya. Retrieved from *https://lrg.upc.edu/en/facilities/ebro-valley/ebro-valley-laboratory*

[6] ASIM. (2019). Retrieved from *https://asim.dk/payload.php*

[7] Wickert, J., Cardellach, E., Martin-Neira, M., Bandeiras, J., Bertino, L., Andersen, O. et al. (2016). GEROS-ISS: GNSS REflectometry, Radio Occultation, and Scatterometry Onboard the International Space Station. IEEE Journal Of Selected Topics In Applied Earth Observations And Remote Sensing, 9(10), 4552-4581. doi: 10.1109/jstars.2016.2614428

[8] ESA. (2019). European Users Guide to Low Gravity Platforms. Retrieved from *http://wsn.spaceflight.esa.int/docs/EuropeanUserGuide/chapter$_{7i}$ss.pdf*

[9] Orbital elements. (2019). Retrieved from *https://en.wikipedia.org/wiki/Orbital$_e$lements*

[10] ISS Telemetry. (2019). Retrieved from *http://www.telemetry.space*

[11] Gómez Tierno, M., Pérez Cortés, M., Puentes Márquez, C. (2012). Mecánica del vuelo. Madrid: Ibergaceta

[12] Calendario gregoriano - EcuRed. (2019). Retrieved from *https://www.ecured.cu/Calendario$_g$regoriano*

[13] Julian Date definition and information. (2019). Retrieved from *https://www.defit.org/julian-date/*

[14] Retrieved from *http://celestrak.com/NORAD/elements/stations.txt*

[15] The Newton-Raphson Method. (2019). Retrieved from *http://www.math.ubc.ca/ anstee/math104/newtonmethod.pdf*

[16] Longitude, U., Rhodes, B. (2019). Using pyephem to calculate when a satellite crosses a Longitude. Retrieved from *https://stackoverflow.com/questions/15338324/using-pyephem-to-calculate-when-a-satellite-crosses-a-longitude*

[17] Media, M. (2019). ISSTracker Space Station Historical Locations. Retrieved from *http://www.isstracker.com/historical*

[18] CartoSat-2 - eoPortal Directory - Satellite Missions. (2019). Retrieved from *https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/cartosat-2*

[19] Shiab, N. (2019). Tutorial: How to send an email with Python - Nael Shiab. Retrieved from *http://naelshiab.com/tutorial-send-email-python/*

[20] Earthdata Login. (2019). Retrieved from *https://ghrc.nsstc.nasa.gov/pub/lis/iss/data/science/nqc/hdf/*

[21] International Atomic Time. (2019). Retrieved from *https://www.timeanddate.com/time/international-atomic-time.html*

[22] iss-appendix-c. (2019). Retrieved from *http://civiliancomms.tripod.com/spacecomms/iss/iss$_a$ppendix$_c$.html*

[23] Earth centered inertial. (2019). Retrieved from *https://en.wikipedia.org/wiki/Earth-centered$_i$nertial*

[24] Cleary, Sean O'Connor, William. (2016). Control of Space Debris Using an Elastic Tether and Wave-Based Control. Journal of Guidance, Control, and Dynamics. 39. 1-15. 10.2514/1.G001624.

[25] Properties, C., Properties, T. (2019). Parcela de contorno de matriz - MATLAB contour- MathWorks España. Retrieved from *https://es.mathworks.com/help/matlab/ref/contour.html*

[26] How do I plot the line of intersection between two surfaces? - MATLAB Answers - MATLAB Central. (2019). Retrieved from *https://es.mathworks.com/matlabcentral/answers/93623-how-do-i-plot-the-line-of-intersection-between-two-surfaces*

[27] coordinates, E., F, M. (2019). Equations to convert from global cartesian coordinates to geographic coordinates. Retrieved from *https://gis.stackexchange.com/questions/120679/equations-to-convert-from-global-cartesian-coordinates-to-geographic-coordinates*

[28] Complemento Georreferenciador. (2019). Retrieved from *https://docs.qgis.org/2.8/es/docs/user$_m$anual/plugins/plugins$_g$eoreferencer.html*

[29] LIS Space Time Domain Search. (2019). Retrieved from *https://lightning.nsstc.nasa.gov/isslisib/isslissearch.html*

[30] Construct map axes for given region of world - MATLAB worldmap- MathWorks España. (2019). Retrieved from *https://es.mathworks.com/help/map/ref/worldmap.html*